UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMÁTICA

Resolución: RPC-SO-02-No.053-2021

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGÍSTER

Título del proyecto:
Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera
Línea de Investigación:
Ciencias de la ingeniería aplicadas a la producción, sociedad y desarrollo sustentable
Campo amplio de conocimiento:
Tecnologías de la Información y la Comunicación (TIC)
Autor/a:
Lino Jesús Cajas Pacheco
Tutor/a:
Mg. Renato Toasa PhD. Maryory Urdaneta

Quito – Ecuador

2024

APROBACIÓN DEL TUTOR

Yo, Renato Toasa con C.I: 1804724167 en mi calidad de Tutor del proyecto de investigación titulado: Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera.

Elaborado por: Lino Jesús Cajas Pacheco, de C.I: 1721061214, estudiante de la Maestría: Seguridad Informática, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., 28 de agosto de 2024

Firmado electrónicamente por:
RENATO MAURICIO
TOASA GUACHI

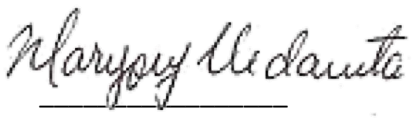
Firma

APROBACIÓN DEL TUTOR

Yo, Maryory Urdaneta con C.I: 1759316126 en mi calidad de Tutor del proyecto de investigación titulado: Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera.

Elaborado por: Lino Jesús Cajas Pacheco, de C.I: 1721061214, estudiante de la Maestría: Seguridad Informática, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., 28 de agosto de 2024



Firma

DECLARACIÓN DE AUTORIZACIÓN POR PARTE DEL ESTUDIANTE

Yo, Lino Jesús Cajas Pacheco con C.I: 1721061214, autor/a del proyecto de titulación denominado: Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera. Previo a la obtención del título de Magister en Seguridad Informática.

1. Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar el respectivo trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Manifiesto mi voluntad de ceder a la Universidad Tecnológica Israel los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor@ del trabajo de titulación, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital como parte del acervo bibliográfico de la Universidad Tecnológica Israel.
3. Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de prosperidad intelectual vigentes.

Quito D.M., 28 de agosto de 2024

_____

Firma

Tabla de contenidos

Índice de tablas

Índice de figuras

INFORMACIÓN GENERAL

Contextualización del tema

El presente proyecto tiene como objetivo crear una guía para la aplicación del "Ciclo de Vida de Desarrollo de Software Seguro" en el proceso de aprendizaje de los estudiantes en desarrollo de software. Esta guía busca enseñar a los alumnos de institutos de educación superior, específicamente en el Instituto Tecnológico Universitario Cordillera, a crear software seguro, utilizando los principales fundamentos del S-SDLC y OWASP como caso de estudio.

El desarrollo de software seguro es considerado como una disciplina que se enfoca en la creación de aplicaciones informáticas que ofrecen protección contra posibles amenazas y vulnerabilidades. En el mundo actual, donde la tecnología desempeña un rol esencial en diversos aspectos de la vida, la seguridad del software se ha vuelto esencial para reforzar la disponibilidad, integridad y privacidad de los datos y sistemas. Así mismo, la importancia de desarrollar aplicaciones seguras y, por consiguiente, la inclusión de medidas de seguridad en las metodologías de desarrollo, se iguala en relevancia a la importancia dada a la seguridad en la construcción de edificios o en la fabricación de barcos (Ortega, 2020).

"El ciclo de vida del desarrollo de software (SDLC)" es un conjunto de prácticas metodológicas empleado en la industria del software para administrar el proceso del desarrollo de aplicaciones, desde su concepción inicial hasta su implementación y mantenimiento posterior. Este ciclo comprende diversas etapas, que incluyen la planificación, el análisis de requisitos, el diseño, la implementación, las pruebas, la entrega y el mantenimiento del software. Además, el SDLC facilita una gestión eficiente del proyecto al proporcionar un marco estructurado para la coordinación de recursos, la asignación de actividades y el seguimiento del progreso del desarrollo.

El marco de referencia S-SDLC es aquel que integra las prácticas de "desarrollo de software" con la "seguridad informática" desde su inicio del "ciclo de vida del desarrollo de software". Este enfoque busca incorporar la seguridad de manera continua y automatizada en todos los procesos del desarrollo, comenzando en la planificación, el diseño hasta la implementación y monitoreo, enmarcando sus principales fundamentos en el: "Modelado de amenazas, casos de abuso, modelado de ataques, ingeniería de requisitos de seguridad, análisis de riesgos arquitectónicos hasta los patrones de diseño".

Según Trujillo y Chávez (2016) "The Open Web Application Security Project" (OWASP) se caracteriza como un proyecto de seguridad de aplicaciones web de código abierto, dedicado a

identificar las raíces del software no seguro. Su enfoque principal radica en proporcionar un listado de las diez vulnerabilidades más graves en aplicaciones web, con la intención primordial de brindar capacitación a diseñadores, desarrolladores, administradores, arquitectos, y organizaciones.

En esta área de investigación, el enfoque se centra en los institutos de educación superior, que se definen como entidades académicas que brindan programas de enseñanza y formación más avanzados que los ofrecidos en los niveles de educación secundaria. Estos establecimientos pueden abarcar una amplia gama de institutos, como universidades, colegios, instituciones técnicas y escuelas de arte, entre otros. Su principal objetivo radica en impartir educación terciaria o postsecundaria en diversas disciplinas académicas y profesionales.

El Instituto Tecnológico Universitario Cordillera (ITSCO) es una empresa dedicada a la educación en diferentes áreas de estudio, una de ellas es la carrera de Desarrollo de Software en la cual se enseña a los alumnos a elaborar sistemas informáticos mediante diferentes metodologías y en el marco del “Ciclo de vida del desarrollo del software” (SDLC), evidenciando en una primera valoración la importancia de la enseñanza y la creación de software seguro a nivel educativo.

Problema de investigación

Los avances tecnológicos impulsados por Internet y sus diversas aplicaciones han dado lugar a una nueva generación que se enfrenta continuamente a desafíos, lo que impulsa el desarrollo de habilidades y mentalidades relacionadas con el uso y la creación de nuevos sistemas digitales. Así mismo, en paralelo a estos avances, han emergido amenazas en diversas formas que resultan cada vez más difíciles de detectar, poniendo en riesgo la seguridad de aquellos usuarios y desarrolladores de software que carecen de la formación necesaria para reconocerlas, neutralizarlas o prevenirlas en diferentes situaciones (Díaz, 2015).

Según lo señalado por Howell (2022), una vulnerabilidad de seguridad se define como una falla o debilidad presente en un sistema de información, lo cual compromete su nivel de seguridad. Es un "agujero" que puede surgir debido a una configuración incorrecta, ausencia de procedimientos, deficiencias en el diseño o por la omisión de medidas de seguridad durante el desarrollo del software. Los ciberdelincuentes utilizan las debilidades presentes en los sistemas informáticos, como las fallas en los sistemas operativos o las deficiencias en el código, para obtener acceso no autorizado y llevar a cabo acciones ilegales, como el robo de información sensible o la interrupción de las operaciones normales del sistema.

En la era digital actual, la prioridad de garantizar la seguridad en el desarrollo de software ha aumentado significativamente, siendo un aspecto cada vez más relevante en la formación universitaria como en el ámbito profesional. Este cambio se da en un contexto en el que la sociedad depende ampliamente de sistemas informáticos en distintos aspectos de la vida cotidiana. Además, el desarrollo de software seguro no solo abarca conocimientos técnicos, sino también una comprensión profunda de los principios de seguridad, el reconocimiento de vulnerabilidades, la integración de medidas de protección y prácticas de seguridad en todas las etapas del “SDLC”.

El enfoque S-SDLC emerge como una respuesta esencial a los desafíos contemporáneos en seguridad cibernética dentro del desarrollo de software. Su integración en la enseñanza en instituciones de educación superior no sólo es relevante, sino imprescindible. Al abordar la seguridad desde los primeros pasos del desarrollo, promueve una cultura proactiva de seguridad que va más allá de simplemente corregir problemas después de que surjan.

Según Diaz (2015) la principal distinción entre la orientación del S-SDLC y los métodos tradicionales radica en que, en los enfoques convencionales, el equipo de seguridad implementa las medidas pertinentes una vez que el producto ha sido completado. Por otro lado, el ciclo de vida del desarrollo de software seguro integra y asegura la implementación de controles de seguridad importantes desde las primeras fases de la creación del software sin comprometer la velocidad de implementación.

El Instituto Tecnológico Universitario Cordillera "ITSCO" es una entidad privada con una trayectoria de más de 31 años en el Ecuador, específicamente en el norte de la ciudad de Quito. Proporciona una variedad de programas educativos de nivel universitario que conducen a la obtención de diversos títulos en distintas áreas de estudio. Una de ellas es la “Carrera de Desarrollo de Software”.

A pesar del éxito de la institución, la carrera de Desarrollo de Software enfrenta el desafío de adaptarse a la creciente y cambiante era digital. Actualmente, el plan académico no incluye un enfoque en técnicas seguras, como el Ciclo de Vida de Desarrollo de Software Seguro (S-SDLC). Es fundamental incorporar nuevas metodologías que aborden la seguridad en todas las etapas del “ciclo de vida del software”. Incluir este enfoque en la formación académica proporcionará a los estudiantes los conocimientos necesarios para enfrentar las diversas vulnerabilidades presentes en los sistemas informáticos que desarrollarán en su vida profesional.

Objetivo general

Elaborar una guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para los estudiantes de la carrera de desarrollo de software del Instituto Tecnológico Universitario Cordillera.

Objetivos específicos

- Contextualizar los fundamentos teóricos sobre el desarrollo de software seguro basado en los principios del S-SDLC para la enseñanza de los estudiantes en la creación de software en el Instituto Tecnológico Universitario Cordillera.
- Diagnosticar las prácticas actuales de la enseñanza del ciclo de vida del software en el Instituto Tecnológico Universitario Cordillera, mediante encuestas a docentes y estudiantes de la carrera de Desarrollo de Software.
- Diseñar una guía para la aplicación de las técnicas del desarrollo de software seguro en el Instituto Tecnológico Universitario Cordillera, basada en los conceptos fundamentales del (S-SDLC).
- Valorar el impacto sobre la aplicación de la enseñanza del software seguro en el Instituto Tecnológico Universitario Cordillera mediante la recopilación y análisis de la opinión de especialistas.

Vinculación con la sociedad y beneficiarios directos:

La creación de una guía para la aplicación del Ciclo de Vida del Software Seguro en la enseñanza en instituciones de educación superior se vincula directamente con el Objetivo de Desarrollo Sostenible (ODS) 4: Educación de Calidad. Al proporcionar una formación avanzada y actualizada en prácticas de desarrollo seguro y eficiente, esta propuesta contribuye a garantizar una educación inclusiva y equitativa de calidad, promoviendo oportunidades de aprendizaje para todos. Además, al preparar a los estudiantes para enfrentar los desafíos tecnológicos contemporáneos, se fortalece la capacidad de los futuros profesionales para innovar y desarrollar soluciones que beneficien a la sociedad, alineándose con el objetivo de promover un desarrollo tecnológico sostenible y responsable.

La vinculación con la sociedad reside en preparar a los estudiantes con competencias avanzadas en seguridad y eficiencia en el desarrollo de software, las instituciones educativas están contribuyendo a la formación de profesionales que pueden diseñar y mantener sistemas seguros y confiables. Esto es particularmente crucial en un mundo cada vez más digitalizado, donde la seguridad de la información es una preocupación prioritaria para gobiernos, empresas

y usuarios finales. Al incorporar un marco de trabajo como el S-SDLC, los graduados estarán mejor equipados para enfrentar desafíos contemporáneos y futuros en el ámbito de la seguridad informática y el desarrollo de software.

Los beneficiarios directos de esta propuesta incluyen, en primer lugar, a los estudiantes de la carrera de Desarrollo de Software del Instituto Tecnológico Universitario Cordillera. Estos estudiantes adquirirán habilidades y conocimientos que los harán altamente competitivos en el mercado laboral. La formación en el S-SDLC no solo mejora su capacidad técnica, sino que también promueve una mentalidad de colaboración y responsabilidad compartida entre desarrollo, operaciones y seguridad. Además, los profesores se benefician de la actualización de sus conocimientos y la adopción de nuevas prácticas educativas que pueden enriquecer su metodología de enseñanza y su carrera profesional.

En un contexto más amplio, las empresas y organizaciones también se benefician al poder contratar profesionales mejor preparados y conscientes de la importancia de la seguridad desde las primeras etapas del desarrollo de software. Esto puede resultar en una reducción de incidentes de seguridad y una mejora en la calidad de los productos tecnológicos. Los usuarios finales, es decir, la sociedad en general, también se ven beneficiados por la existencia de aplicaciones y servicios más seguros y confiables. Al final, esta iniciativa fortalece el ecosistema tecnológico, fomenta la innovación y contribuye al bienestar general al proporcionar soluciones tecnológicas que protegen mejor la información y los datos de los usuarios.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1.1. Contextualización general del estado del arte

La seguridad en el desarrollo de software se ha convertido en una prioridad para el Instituto Tecnológico Universitario Cordillera, dado el creciente número de ciberataques y vulnerabilidades en sistemas de diferentes entornos. En este contexto, el marco de trabajo S-SDLC, integra prácticas de desarrollo y seguridad, se presenta como una solución innovadora para garantizar la protección integral a lo largo del ciclo de vida del desarrollo del software. El Ciclo de Vida de Desarrollo de Software Seguro promueve una cultura de colaboración entre equipos de desarrollo y seguridad, automatizando procesos de seguridad y garantizando la incorporación de controles desde las primeras etapas del desarrollo.

Contextualización

El desarrollo de una guía para la aplicación de medidas de seguridad en la enseñanza del SDLC garantizará que los alumnos del Instituto Cordillera creen sistemas seguros. Esta integración es esencial para preparar a los estudiantes con las competencias necesarias para enfrentar desafíos de seguridad en entornos de desarrollo modernos y dinámicos. La ciberseguridad no es solo responsabilidad de los departamentos de TI o de seguridad; es fundamental una colaboración integral en la carrera de Desarrollo de Software de la institución para abordar los desafíos relacionados con la integración de seguridades en el ciclo de vida del desarrollo de software que se les enseña a los estudiantes.

Conceptos fundamentales

Desarrollo de Software

De acuerdo con IBM (2024) el desarrollo de software implica una serie de actividades informáticas orientadas a la creación, diseño, implementación y soporte del software. En esencia, el desarrollo de software es un proceso lógico e iterativo destinado a crear programas informáticos. Este proceso incluye varias etapas, como la recolección de información, el diseño de flujos, la documentación, pruebas, y la depuración, entre otras.

Software Development Life Cycle (SDLC)

El “Ciclo de Vida del Desarrollo de Software” (SDLC) representa un procedimiento fundamental en la creación del software, abarcando desde la idea inicial de una aplicación hasta su puesta en marcha y soporte continuo. Estas fases comprenden la planificación, diseño, desarrollo, pruebas, implementación y el mantenimiento del software. En la actualidad,

enfoques como DevOps y S-SDLC tienen como objetivo agilizar y afianzar el ciclo de vida, asegurando la calidad y la integración de seguridades del software en la totalidad de sus etapas (Pachacuti, 2020).

Según Sierra (2022) menciona que:

> El ciclo de vida de desarrollo de software contiene todas las fases del desarrollo desde el inicio de la idea hasta su funcionamiento. Cada fase es un engranaje fundamental de la siguiente fase por lo cual cada fase tiene la misma importancia en todo el proceso (p. 54).

Figura 1.
Ciclo de vida de un software

Nota. El ciclo de vida de un software (Álvarez, 2018).

Seguridad en el Desarrollo de Software

La Seguridad en el Desarrollo de Software se define como el conjunto de prácticas, técnicas y principios implementados durante todas las etapas del ciclo de vida del desarrollo de software para proteger las aplicaciones y sistemas contra vulnerabilidades, amenazas y ataques.

(López, 2020) menciona que:

> Dentro del ciclo de vida del software se puede entender por seguridad los requerimientos del sistema y poder cumplir con sus funciones, pero realmente no es así, se estima que hay una gran confusión con el tema seguridad el cual es un concepto delicado con respecto al manejo de datos. Es importante aplicar la seguridad al ciclo de vida del software con el fin de tener calidad en los productos desarrollados y cumplir con los tres objetivos de la seguridad que son la integridad, confidencialidad y disponibilidad (p. 264).

Seguridad informática

(López, 2020) subraya que la seguridad informática debe integrarse estrechamente con la "Ingeniería de Software" para asegurar que el desarrollo del sistema se mantenga en condiciones óptimas. Esto es fundamental no solo para cumplir con los requisitos tecnológicos en constante evolución, sino también para satisfacer las necesidades de los usuarios y proveedores que utilizan estos sistemas. Esta integración es crucial para garantizar la protección contra amenazas y vulnerabilidades, así como para asegurar que los sistemas puedan adaptarse y responder adecuadamente a nuevos desafíos y demandas del mercado. Además, refuerza la importancia de una colaboración continua entre los equipos de desarrollo y seguridad para lograr un software robusto y confiable.

Principios de Seguridad

Según Pérez et al. (2023) menciona que los principios de seguridad pueden orientar el diseño y la implementación de software sin vulnerabilidades. Si durante el desarrollo no se consideran estos principios, o se incumplen, ello puede indicar una posible falla en el proceso. Por lo tanto, es crucial revisar minuciosamente el proceso de desarrollo para controlar y corregir cualquier fallo. Microsoft se refiere a estos principios como SD3+C: Seguro por diseño, Seguro por defecto, Seguro en implementación y Comunicaciones. Las breves definiciones de estos principios son:

- Seguro por Diseño: El software debe ser construido, diseñado e implementado para resistir ataques y proteger tanto a sí mismo como a la información que procesa.
- Seguro por Defecto: Para reducir el daño cuando un atacante explota una vulnerabilidad, la configuración predeterminada del software debe optar por las opciones más seguras.
- Seguro en el Despliegue: El software debe venir acompañado de información y herramientas que ayuden a los administradores y usuarios a usarlo de manera segura. Las actualizaciones deben ser fáciles de distribuir y debe haber un sistema de respuesta rápida ante nuevas amenazas.
- Seguro en las Comunicaciones: El equipo de desarrollo debe estar preparado para identificar vulnerabilidades de seguridad en el producto y comunicarse de manera abierta y responsable con los usuarios.

Microsoft SDL

Pérez et al. (2023) resalta que "El ciclo de vida de desarrollo de seguridad (SDL) es un proceso de control de seguridad orientado al desarrollo de software" (p. 137). Microsoft SDL introduce

la seguridad y la privacidad en todas las fases del proceso de desarrollo, fases que contienen un conjunto de actividades de carácter obligatorio.

Figura 2.
Ciclo de vida del software de Microsoft

Nota. Ciclo de vida de desarrollo de software de Microsoft: simplificado (Microsoft, 2010).

Amenazas y vulnerabilidades del software

En el contexto del desarrollo de software, la información puede estar expuesta a riesgos de daño o pérdida. El aumento en la demanda de interconexión a nivel global ha intensificado las amenazas y vulnerabilidades en los sistemas y redes de información, exponiéndose a un número cada vez mayor y más variado de amenazas, esta situación subraya la necesidad de fortalecer las medidas de seguridad en todas las etapas del ciclo de vida del software (Parraga, 2024). Además, la evolución constante de las tecnologías y las técnicas de ataque requiere una actualización continua de las prácticas de seguridad para mitigar los riesgos asociados (López, 2020).

1.2. Proceso investigativo metodológico

El proceso metodológico de esta investigación se basa en un enfoque cuantitativo, que emplea estrategias investigativas centradas en la recopilación de datos numéricos a través de encuestas. Según Sampieri (2014), este enfoque permite obtener información cuantificable, facilitando un análisis objetivo de la situación. En este caso, el objetivo es contextualizar y evaluar el estado actual del conocimiento y la integración de la seguridad en el currículo académico de la “Carrera de Desarrollo de Software del Instituto Tecnológico Universitario Cordillera”.

A través de la recopilación y análisis de datos, se busca medir la percepción y preparación de estudiantes y docentes en relación con prácticas de desarrollo de software seguro, proporcionando una visión general de cómo se abordan estos temas en la institución.

Se emplearon tanto métodos teóricos como prácticos en este estudio. Los métodos teóricos incluyeron la revisión de literatura y teorías sobre seguridad en el desarrollo de software, así como el marco de trabajo S-SDLC. En cuanto a los métodos prácticos se realizaron encuestas dirigidas a estudiantes y docentes como herramienta principal para la recolección de datos. Estas encuestas fueron diseñadas para obtener respuestas cerradas, permitiendo así un análisis estadístico de los resultados que fueron analizados para obtener una visión comprehensiva del estado actual de la enseñanza y la aplicación de la seguridad en el desarrollo de software.

El tipo de investigación fue descriptivo, lo que permitió analizar las características del fenómeno para definir, clasificar, dividir o resumir. Este tipo de investigación suele ser una etapa preliminar antes de los diseños de investigación cuantitativa, y en este caso, ayudó a conocer el nivel de enseñanza en seguridad en el desarrollo de software entre los estudiantes, así como la falta de inclusión del enfoque S-SDLC en el currículo académico.

El universo de estudio estuvo constituido por los estudiantes y docentes de la "Carrera de Desarrollo de Software del Instituto Tecnológico Universitario Cordillera", quienes accedieron a participar voluntariamente en este proyecto bajo consentimiento informado. Esta participación facilitó la propuesta de una guía de estudio para la mejora de la enseñanza sobre seguridad en el desarrollo de software, abarcando tanto la evaluación del conocimiento de los estudiantes como la percepción de los docentes respecto a la inclusión de un marco de trabajo como el S-SDLC en el currículo académico.

Según Sampieri (2014) menciona que "Las muestras no probabilísticas, también llamadas muestras dirigidas, suponen un procedimiento de selección orientado por las características de la investigación, más que por un criterio estadístico de generalización" (p. 189). En base a este enfoque la población del estudio incluyó a los estudiantes y docentes de la carrera de Desarrollo de Software. Por lo cual se utilizó un muestreo no probabilístico de tipo intencional para seleccionar a los participantes en las encuestas, asegurando una representación adecuada de los últimos niveles académicos y áreas de especialización como la programación. A continuación, se presenta una tabla con el número total de la población y el tamaño de la muestra:

Tabla 1.

Población y Muestra

Grupo	Población Total	Muestra
Estudiantes	400	30

Docentes	8	6

Nota. La tabla presenta la población y muestra del estudio: de 400 estudiantes, se seleccionaron 40, y de 8 docentes, se seleccionaron 6.

La metodología de trabajo se desarrolló en varias etapas. Primero, se diseñaron y validaron los instrumentos de recolección de datos, garantizando que las preguntas fueran claras, precisas y alineadas con los objetivos del estudio. Luego, se realizaron encuestas estructuradas utilizando "Google Forms", una plataforma online que facilita la captura y análisis de datos para generar tablas y gráficos estadísticos. Estas encuestas se aplicaron a estudiantes y profesores de la carrera de Desarrollo de Software, permitiendo la recolección de datos representativos. Los datos fueron analizados con técnicas estadísticas para identificar tendencias y correlaciones. Finalmente, se formularon conclusiones y recomendaciones orientadas a la creación de una guía para implementar prácticas de seguridad en el desarrollo de software en la carrera.

1.3. Análisis de resultados

Siguiendo la aplicación de los métodos previamente descritos, se procedió a la recolección y sistematización de los datos obtenidos, los cuales se presentan en las siguientes matrices y gráficos detallados a continuación.

1.3.1. Análisis de la encuesta realizada a los estudiantes.

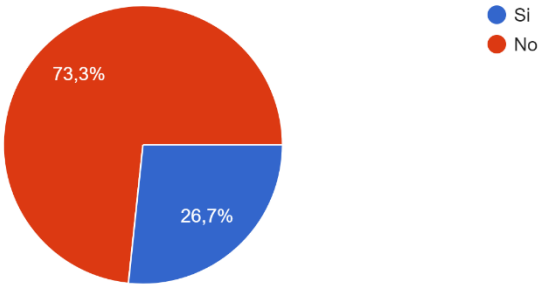
Pregunta 1. Durante su formación académica, ¿se le ha mencionado sobre el concepto de "Ciclo de Vida del Desarrollo de Software Seguro (S-SDLC)"?

Tabla 2.
Conocimiento del S-SDLC

Opciones	Frecuencia	Porcentaje
Si	8	73.3%
No	22	26.7%
Total	30	100%

Nota. Información recopilada a partir de la encuesta realizada a los alumnos de la carrera de Desarrollo de Software

Figura 3.
Conocimiento del S-SDLC



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

En conformidad a la formulación sobre; ¿si durante su formación académica, se les ha mencionado sobre el concepto de "Ciclo de Vida del Desarrollo de Software Seguro (S-SDLC)"?, se evidenció que: El 73.3% (n=22) de los encuestados indicaron que no se les ha mencionado este concepto durante su formación académica. Esto sugiere una carencia significativa en la inclusión de prácticas de desarrollo de software seguro dentro del currículo educativo. Por otro lado, un 26.7% (n=8) de los estudiantes indicaron que sí se les ha mencionado el S-SDLC durante su formación. Aunque este porcentaje es menor, muestra que hay una base inicial de conocimiento sobre este importante tema, lo cual es positivo y podría ser ampliado.

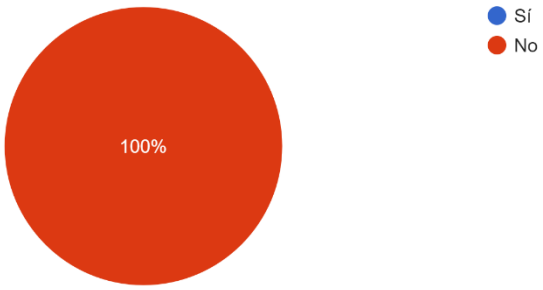
Pregunta 2. Durante su formación académica, ¿se le ha instruido sobre cómo aplicar revisiones de código estáticas y dinámicas, el uso de entornos aislados (Sandbox), y pruebas de penetración (pentesting)?

Tabla 3.
Pruebas de Código

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	30	100%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de la carrera de Desarrollo de Software

Figura 4.
Pruebas de Código



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

Respecto a la a interrogante, se demostró que: El 100% (n=30) de los alumnos encuestados indicaron que no han recibido instrucción en revisiones de código estáticas, dinámicas, el uso de entornos aislados (Sandbox), y pruebas de penetración (pentesting) durante su formación académica.

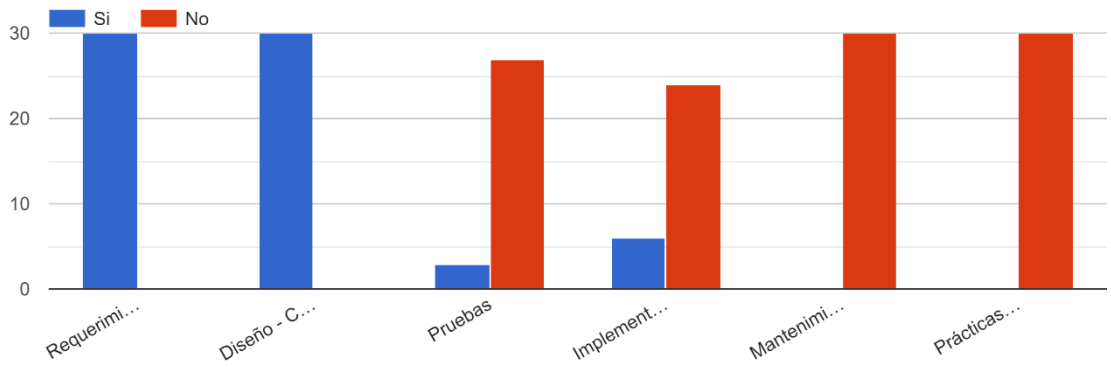
Pregunta 3. ¿En el aprendizaje de su carrera ha cumplido con todos los procesos del Ciclo de Vida de Desarrollo de Software (SDLC) tales cómo?

Tabla 4.
Etapas del SDLC

Etapas del SDLC	Cumplimiento	
	Si	No
Requerimientos	100%	0%
Diseño - Codificación	100%	0%
Pruebas	10%	90%
Implementación (publicar el sistema en un servidor local, público)	20%	80%
Mantenimiento	0%	100%
Prácticas de seguridad en cada una de las fases del ciclo de vida del software	0%	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 5.
Etapas del SDLC



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

Respecto a la interrogante, se demostró que: el cumplimiento de las distintas etapas del Ciclo de Vida del Desarrollo de Software (SDLC) según la percepción de los alumnos encuestados. Se observa que el 100% ha cumplido con las fases de levantamiento de requerimientos, diseño y codificación. Sin embargo, solo el 10% ha realizado pruebas y un 20% ha llevado a cabo la implementación del sistema en un servidor local o público. La etapa de mantenimiento no ha sido cumplida por ningún encuestado (0%). Además, ninguna de las fases del ciclo de vida del software ha incluido prácticas de seguridad, ya que el 100% indica que no se han implementado estas prácticas.

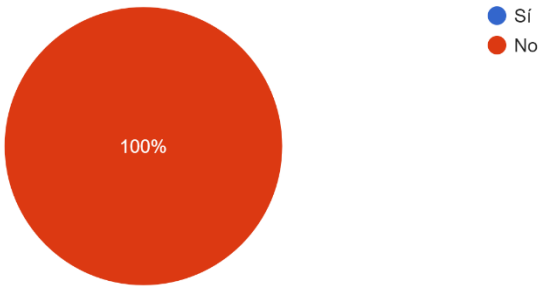
Pregunta 4. En tu opinión, ¿Conoces si en los planes analíticos (PA) de las materias de programación incluyen alguna metodología para el desarrollo de software seguro?

Tabla 5.
Contenido de los planes analíticos de las materias de programación

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	30	100%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de la carrera de Desarrollo de Software

Figura 6.
Contenido de los planes analíticos de las materias de programación



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

Según la pregunta sobre si en los planes analíticos (PA) de las materias de programación se incluye alguna metodología para el desarrollo de software seguro, se observa que el 100% de los encuestados indicó que no tienen conocimiento de la inclusión de tales metodologías.

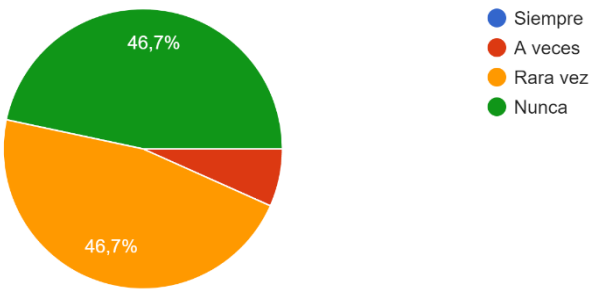
Pregunta 5. ¿Utilizas prácticas de codificación segura en tus proyectos o deberes de las materias de programación?

Tabla 6.
Seguridad en proyectos o deberes de programación

Opciones	Frecuencia	Porcentaje
Siempre	0	0%
A veces	2	6.6%
Rara vez	14	46.7%
Nunca	14	46.7%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 7.
Seguridad en proyectos o deberes de programación



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

De acuerdo con la formulación, se evidenció que, el uso de prácticas de codificación segura en proyectos o deberes de las materias de programación, se evidencia que ninguno de los encuestados las utiliza siempre (0%). Solo el 6.6% (n=2) mencionó que a veces emplean estas prácticas, mientras que el 46.7% (n=14) las utiliza rara vez y otro 46.7% (n=14) indicó que nunca las emplea. Este análisis muestra que una gran mayoría de los estudiantes rara vez o nunca aplican prácticas de codificación segura en sus trabajos académicos, lo cual subraya una considerable deficiencia en la adopción de estas prácticas esenciales para el desarrollo de software seguro.

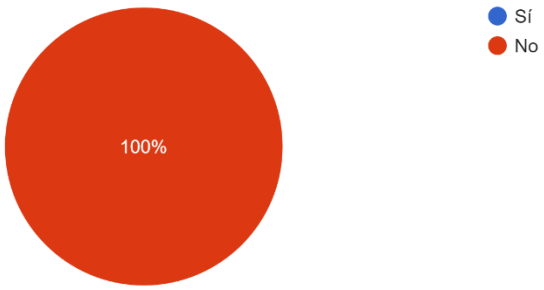
Pregunta 6. ¿Has participado en alguna revisión de código enfocada en la seguridad en algún nivel de la carrera?

Tabla 7.
Práctica o taller de seguridad del software en la carrera

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	30	100%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 8.
Práctica o taller de seguridad del software en la carrera



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

En cuanto a la participación en revisiones de código enfocadas en la seguridad durante la carrera, el 100% de los encuestados (n=30) indicaron que no han tenido esta experiencia. Esto revela una falta total de exposición a revisiones de código con un enfoque en la seguridad en el ámbito académico.

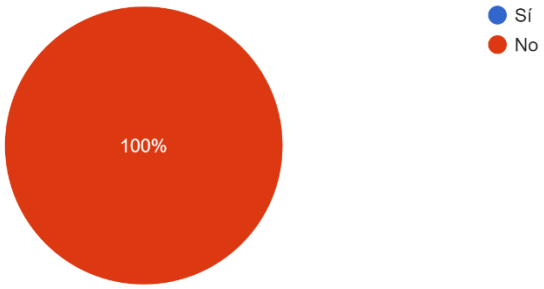
Pregunta 7. ¿Te sientes preparado para aplicar medidas de seguridad en tus proyectos de desarrollo de software fuera del ámbito académico?

Tabla 8.
Aplicabilidad de seguridades del software a nivel profesional

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	30	100%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 9.
Aplicabilidad de seguridades del software a nivel profesional



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

Según la pregunta sobre si los estudiantes se sienten preparados para aplicar medidas de seguridad en sus proyectos de desarrollo fuera del ámbito académico, el 100% de los encuestados (n=30) respondieron que no. Esto indica que ninguno de los estudiantes se siente confiado en su capacidad para implementar medidas de seguridad en contextos profesionales.

Este dato resalta una deficiencia significativa en la formación académica respecto a la preparación para enfrentar desafíos de seguridad en el desarrollo de software en el mundo real.

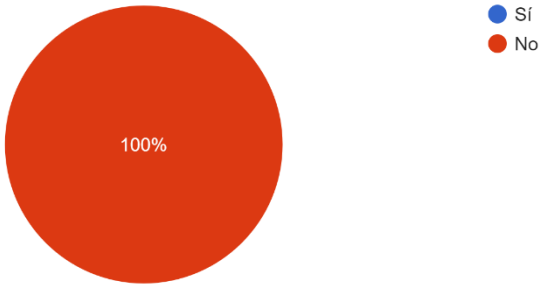
Pregunta 8. ¿Consideras que el plan de estudios actual de la carrera incluye suficientes contenidos sobre seguridad en el desarrollo de software?

Tabla 9.
Opinión sobre el plan de estudios de las materias de programación

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	30	100%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 10.
Opinión sobre el plan de estudios de las materias de programación



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

En cuanto a la pregunta sobre si el plan de estudios actual de la carrera incluye suficientes contenidos sobre seguridad en el desarrollo de software, el 100% de los encuestados (n=30) respondió que no, que el currículo académico carece de suficientes contenidos relacionados con la seguridad en el desarrollo de software.

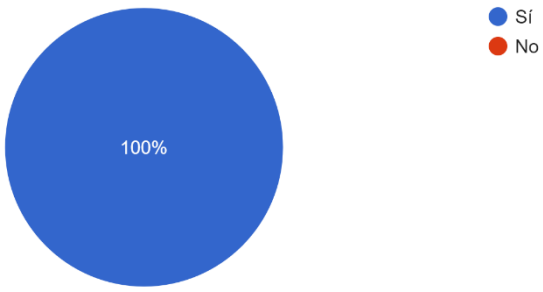
Pregunta 9. ¿Crees que la seguridad debería ser un tema transversal en todos los cursos relacionados con el desarrollo de software?

Tabla 10.
Opinión sobre la integración de seguridad en el desarrollo del software

Opciones	Frecuencia	Porcentaje
Si	30	100%
No	0	0%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 11.
Opinión sobre la integración de seguridad en el desarrollo del software



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

Respecto a la formulación anterior, todos los encuestados (n=30) respondieron afirmativamente, alcanzando un consenso del 100%. Evidenciando una fuerte percepción de la importancia de integrar la seguridad en todas las etapas del aprendizaje del desarrollo del sistema, reconociendo así la necesidad de que la seguridad sea un componente integral y continuo en su formación académica.

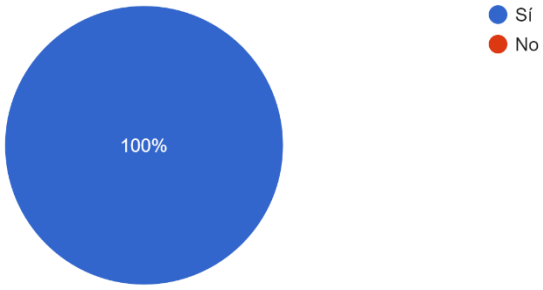
Pregunta 10. ¿Considera que el aprendizaje de prácticas de seguridad en el desarrollo de software es importante para su carrera profesional?

Tabla 11.
Prácticas de seguridad para su carrera profesional como desarrolladores

Opciones	Frecuencia	Porcentaje
Si	30	100%
No	0	0%
Total	30	100%

Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Figura 12.
Prácticas de seguridad para su carrera profesional como desarrolladores



Nota. Información obtenida a partir de la encuesta realizada a los alumnos de Desarrollo de Software.

Análisis

En conformidad a esta formulación, se evidencio que el 100% (n=30) de los encuestados si consideran que el aprendizaje de prácticas de seguridad en el "desarrollo de software" es importante para su carrera profesional. Este consenso absoluto subraya la percepción de que las prácticas de seguridad son esenciales para su futuro profesional, reconociendo así que el conocimiento y la aplicación de medidas de seguridad en el desarrollo de software son fundamentales para enfrentar los desafíos actuales en la industria tecnológica y asegurar la integridad y protección de los sistemas y datos.

Análisis general de la encuesta aplicada a los alumnos

En general, se observa que la formación de los estudiantes carece de enseñanza en seguridad en el desarrollo de software. Muy pocos han recibido instrucción sobre revisiones de código, entornos aislados, pruebas de penetración o metodologías de desarrollo seguro. Todos los encuestados señalaron que el plan de estudios es insuficiente en cuanto a contenido de seguridad y que estos temas deberían integrarse en todos los cursos. Además, expresaron no sentirse preparados para aplicar medidas de seguridad fuera del ámbito académico. Estos hallazgos subrayan la necesidad urgente de mejorar la educación en seguridad del software dentro del currículo académico para formar mejor a los futuros profesionales en desarrollo de software.

1.3.2. Análisis de la encuesta realizada a los docentes.

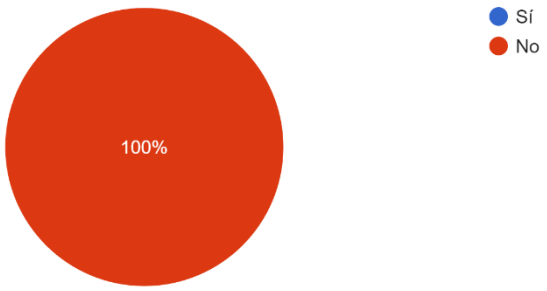
Pregunta 1. ¿Incluye temas de seguridad en el desarrollo de software en sus asignaturas de programación?

Tabla 12.
Integración de seguridades en la enseñanza del Software

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	6	100%
Total	6	100%

Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Figura 13.
Integración de seguridades en la enseñanza del Software



Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Análisis

En conformidad con la formulación sobre si los docentes de programación incluyen temas de "seguridad en el desarrollo de software" en sus asignaturas, se evidenció que el 100% (n=6) de los encuestados indicaron que no los incluyen. Este resultado revela una falta total de integración de temas de seguridad en las asignaturas de programación impartidas por los docentes encuestados.

Pregunta 2. ¿Utiliza herramientas de análisis de seguridad de código como OWASP ZAP, Syhunt, Wireshark, Fortify, SonarQube, etc., en sus clases de programación?

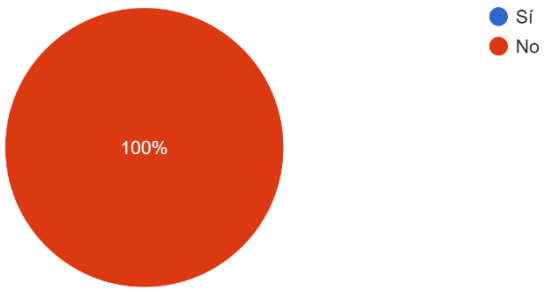
Tabla 13.
Uso de herramientas de análisis de seguridad

Opciones	Frecuencia	Porcentaje
Si	0	0%

No	6	100%
Total	6	100%

Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Figura 14.
Uso de herramientas de análisis de seguridad



Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Análisis

Respecto a la formulación sobre si los docentes de programación utilizan herramientas de análisis de seguridad de código en sus clases de programación, todos los encuestados (n=6) respondieron negativamente, alcanzando un consenso del 100%. Evidenciando una falta de empleo de herramientas críticas para el análisis de seguridad del código en la enseñanza de programación.

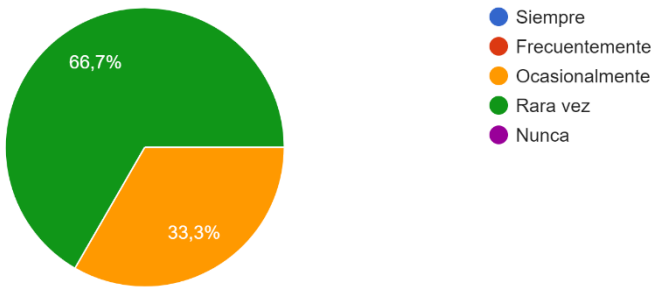
Pregunta 3. ¿Con qué frecuencia discute sobre seguridad en el desarrollo de software en sus clases?

Tabla 14.
Temas de Seguridad del Software en las clases

Opciones	Frecuencia	Porcentaje
Siempre	0	0%
Frecuentemente	0	0%
Ocasionalmente	2	33.3%
Rara vez	4	66.7%
Nunca	0	0%
Total	30	100%

Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Figura 15.
Temas de Seguridad del Software en las clases



Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Análisis

Respecto a la formulación sobre la frecuencia con la que los docentes discuten sobre seguridad en el desarrollo de software en sus clases, se evidenció que el 33.3% (n=2) de los encuestados indicó que lo hace ocasionalmente, mientras que el 66.7% (n=4) respondió que rara vez abordan este tema. Demostrando que la mayoría de los docentes no prioriza las discusiones sobre seguridad en sus clases, lo cual sugiere una falta de énfasis en la enseñanza de prácticas de seguridad esenciales en el desarrollo del software.

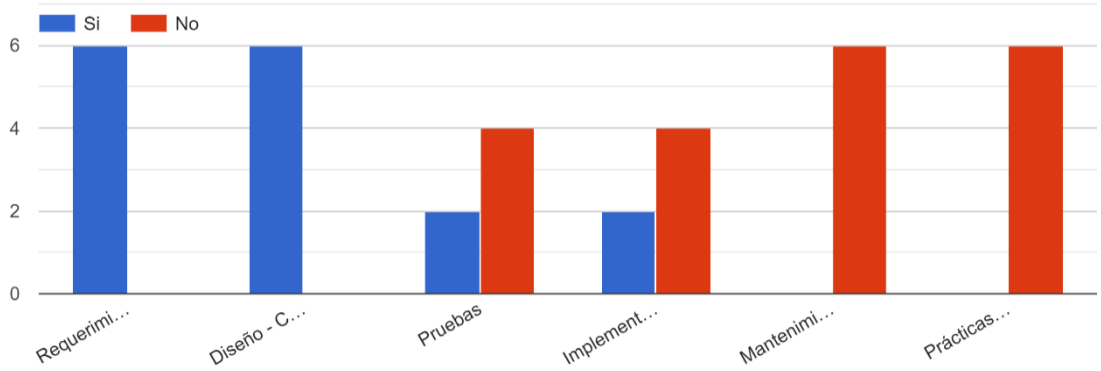
Pregunta 4. ¿En la enseñanza de su materia de programación cree usted que ha cumplido con todos los procesos del Ciclo de Vida de Desarrollo de Software (SDLC) cómo?

Tabla 15.
Etapas de la enseñanza del SDLC

Etapas del SDLC	Cumplimiento	
	Si	No
Análisis de requisitos (Requerimientos)	100%	0%
Diseño - Codificación	100%	0%
Implementación (publicar el sistema en un servidor local, público)	33.3%	66.7%
Pruebas	33.3%	66.7%
Mantenimiento	0%	100%
Prácticas de seguridad en cada una de las fases del ciclo de vida del software	0%	100%

Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Figura 16.
Etapas de la enseñanza del SDLC



Nota. Información recopilada a partir de la encuesta realizada a los docentes de programación.

Análisis

Respecto a la interrogante, se demostró que: el cumplimiento de las distintas etapas del Ciclo de Vida del Desarrollo de Software (SDLC) según los docentes encuestados. En todas las etapas analizadas, el 100% de los docentes cumplen con las fases de levantamiento de requerimientos, diseño y codificación. Sin embargo, solo el 33.3% de los docentes han realizado las pruebas y la implementación del sistema en un servidor local o público. Ninguno de los docentes (0%) se ocupa del mantenimiento ni de la integración de prácticas de seguridad en cada una de las fases del ciclo de vida del software.

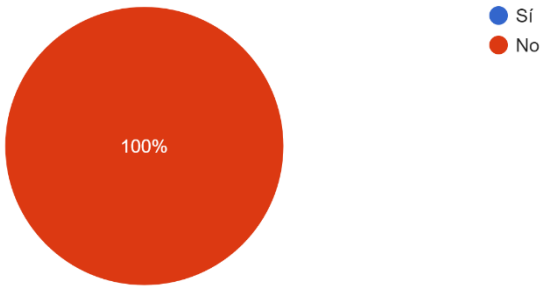
Pregunta 5. ¿Cree que la seguridad es adecuadamente abordada en el actual plan de estudios de la carrera de Desarrollo de Software?

Tabla 16.
Seguridades en el plan de estudio

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	6	100%
Total	6	100%

Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Figura 17.
Seguridades en el plan de estudio



Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Análisis

Respecto a la formulación sobre si la seguridad es adecuadamente abordada en el actual plan de estudios de la carrera de Desarrollo de Software, todos los docentes encuestados (n=6) respondieron negativamente, alcanzando un consenso del 100%. Esto evidencia una deficiencia en el tratamiento de temas de seguridad en el plan de estudios.

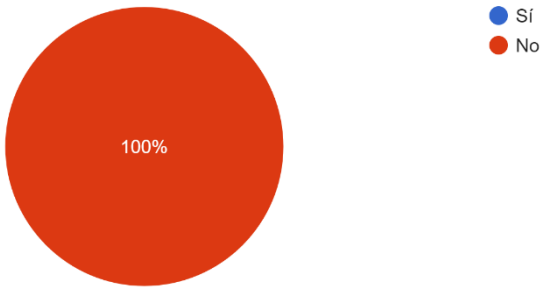
Pregunta 6. ¿Considera que sus estudiantes están adquiriendo suficientes habilidades para desarrollar software seguro?

Tabla 17.
Conocimiento adquirido por estudiantes en seguridad de software

Opciones	Frecuencia	Porcentaje
Si	0	0%
No	6	100%
Total	6	100%

Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Figura 18.
Conocimiento adquirido por estudiantes en seguridad de software



Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Análisis

Según la pregunta sobre si los estudiantes están adquiriendo suficientes habilidades para desarrollar software seguro, todos los docentes encuestados (n=6) respondieron negativamente, alcanzando un consenso del 100%. Manifestando una preocupación común sobre la insuficiencia de la formación en seguridad que reciben los estudiantes.

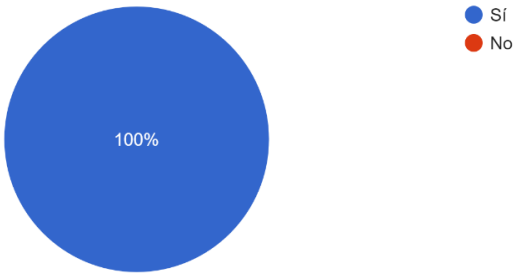
Pregunta 7. ¿Cree que debería haber más enfoque en la seguridad en la formación de los docentes de programación?

Tabla 18.
Formación en seguridades en desarrollo de software a docentes.

Opciones	Frecuencia	Porcentaje
Si	6	100%
No	0	0%
Total	6	100%

Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Figura 19.
Formación en seguridades en desarrollo de software a docentes.



Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Análisis

Respecto a la formulación sobre si los docentes creen que debería haber más enfoque en la seguridad en la formación de los docentes de programación, todos los encuestados (n=6) respondieron afirmativamente, alcanzando un consenso del 100%. Demostrando una fuerte percepción de la importancia de aumentar el enfoque en la seguridad dentro de la formación de los docentes de programación.

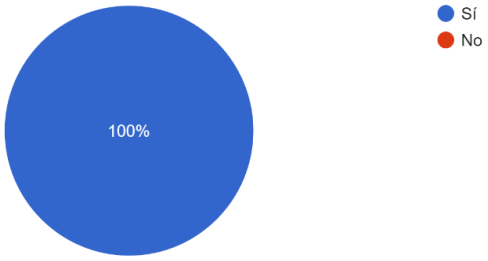
Pregunta 8. ¿Estaría interesado en recibir formación adicional o participar en talleres sobre desarrollo de software seguro?

Tabla 19.
Capacitación en seguridad del software

Opciones	Frecuencia	Porcentaje
Si	6	100%
No	0	0%
Total	6	100%

Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Figura 20.
Capacitación en seguridad del software



Nota. Información obtenida a partir de la encuesta realizada a los docentes de programación.

Análisis

Respecto a la formulación sobre si los docentes estarían interesados en recibir formación adicional o participar en talleres sobre desarrollo de software seguro, todos los encuestados (n=6) respondieron afirmativamente, alcanzando un consenso del 100%. Enmarcando un interés unánime entre los docentes en mejorar sus conocimientos y habilidades en el ámbito de la seguridad en el desarrollo de software.

Análisis general de la encuesta aplicada a los docentes

En general, se observa que la encuesta aplicada a los docentes revela varias deficiencias significativas y áreas de mejora en la formación y enseñanza de la “seguridad en el desarrollo de software”. Ninguno de los docentes incluye temas de seguridad en sus asignaturas de programación ni utiliza herramientas de análisis de seguridad de código, como OWASP ZAP o SonarQube, entre otras. Además, solo un pequeño porcentaje discute ocasionalmente sobre seguridad en sus clases. Aunque una mayoría significativa está familiarizada con las amenazas comunes de seguridad, ninguno enseña a los estudiantes cómo protegerse contra estas amenazas. Todos los encuestados coinciden en que debería haber un mayor enfoque en la seguridad dentro de la formación docente y muestran un interés unánime en recibir formación adicional o participar en talleres sobre desarrollo de software seguro.

CAPÍTULO II: PROPUESTA

2.1. Fundamentos teóricos aplicados

Secure Software Development Life Cycle (S-SDLC)

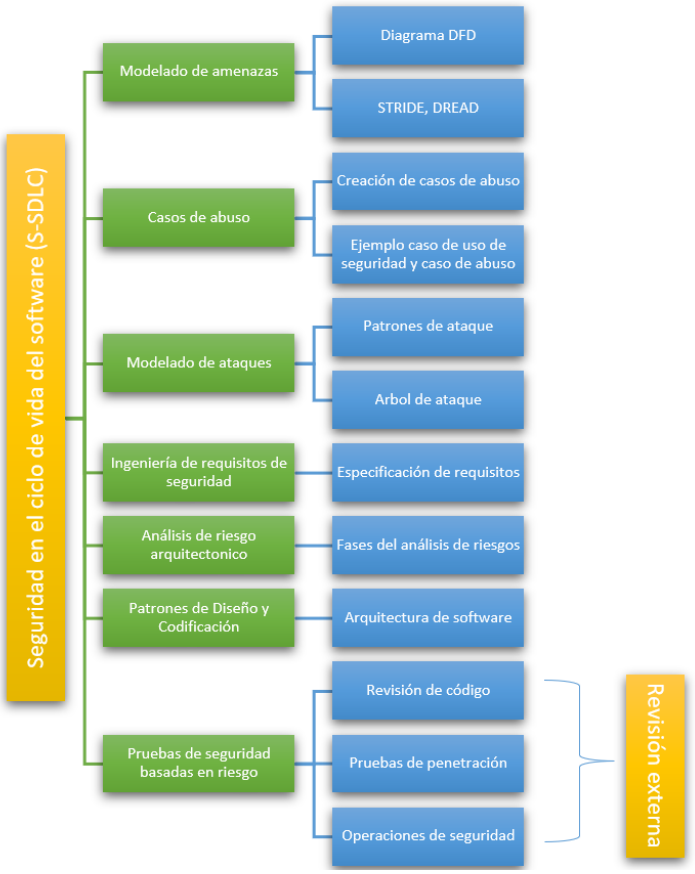
El “ciclo de vida del desarrollo de software seguro” define las fases o procesos que un software debe atravesar para garantizar su solidez, cumplir con las necesidades del usuario final y asegurar la creación de un producto de alta calidad. Según Trujillo y Chávez (2016) hace mención que buscar ofrecer al cliente un sistema completamente seguro puede ser una meta difícil de alcanzar y un tanto idealista, dado que, aunque las técnicas de seguridad han avanzado, la tecnología también ha permitido a individuos malintencionados desarrollar métodos más sofisticados para comprometer el software final.

Fundamentos generales del desarrollo de software seguro

La creación de “software seguro” y confiable requiere adoptar un enfoque sistemático que integre la seguridad en cada etapa del ciclo de vida del desarrollo del software (Viteri, 2023).

> Según López (2020) menciona que en el desarrollo de software seguro:
>
> Se debe integrar en él dos tipos de actividades de seguridad: la primera es el seguimiento de unos principios de diseño seguro (mínimo privilegio, por ejemplo); y la segunda, la inclusión de una serie de buenas prácticas de seguridad (especificación de requisitos de seguridad, casos de abuso, análisis de riesgo, análisis de código, pruebas de penetración dinámicas, etc.) (p. 4).

Figura 21.
Fundamentos del S-SDLC



Nota. Principales fundamentos de la seguridad del software Basada en (López, 2020)

La seguridad en el ciclo de vida del software

Según López (2020) menciona que "La seguridad del software es algo más que la eliminación de las vulnerabilidades y la realización de pruebas de penetración." (p. 6). Un punto crucial es que los gerentes adopten un enfoque integral que incorpore principios de diseño y buenas prácticas de seguridad en el ciclo de vida del desarrollo del software. El objetivo es crear software más seguro y confiable y garantizar su seguridad a través de verificaciones constantes.

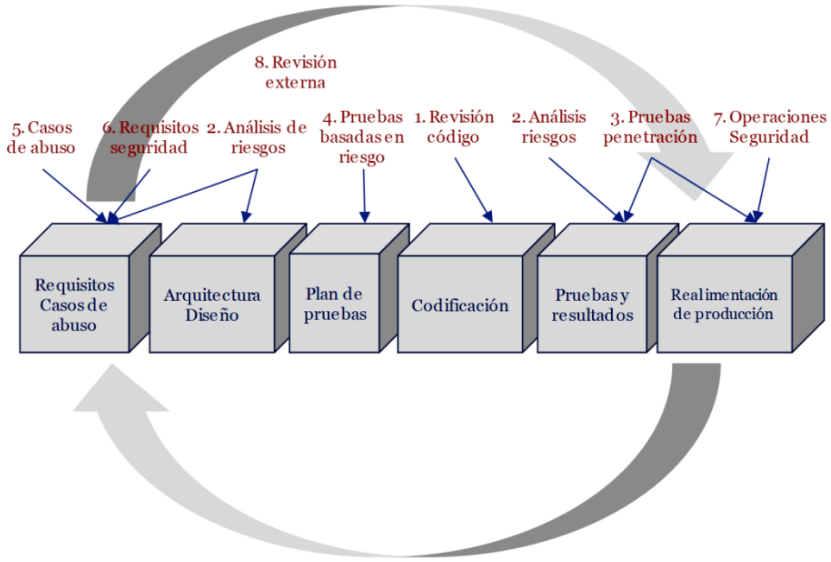
Este enfoque integral en el ciclo de vida del desarrollo se denomina Ciclo de Vida del Software Seguro (S-SDLC).

Siete puntos de contacto para la seguridad del software

McGraw (2006) menciona que los siete puntos de contacto ofrecen una serie de mejores prácticas organizadas que se pueden aplicar a diversos artefactos del desarrollo de software, sin estar restringidas a un ciclo de desarrollo específico. Estas prácticas incluyen tanto actividades constructivas (como diseño, defensa y funcionalidad) como destructivas (como ataques y exploits). Los puntos de contacto se presentan en un orden basado en su efectividad.

1. Revisión de código.
2. Análisis de riesgo arquitectónico.
3. Pruebas de penetración.
4. Pruebas de seguridad basadas en riesgos.
5. Casos de abuso.
6. Requerimientos de seguridad.
7. Operaciones de seguridad.

Figura 22.
Siete puntos de contacto

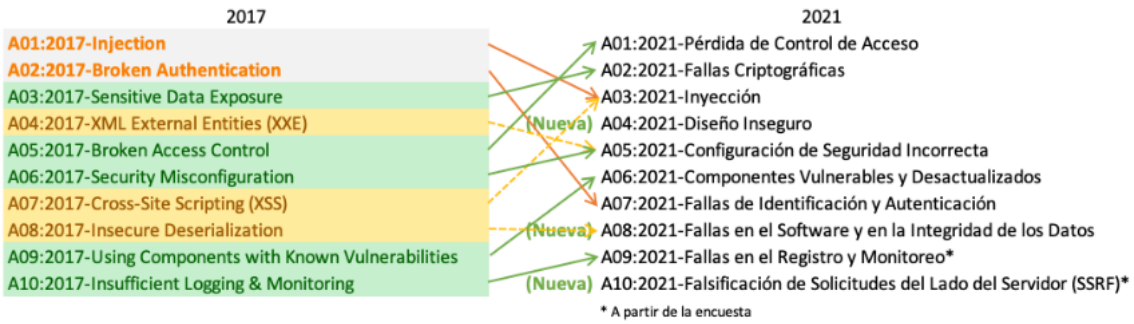


Nota. Puntos de contacto de seguridad del software (McGraw, 2006).

OWASP (Open Web Application Security Project)

Según Ortega (2020) en su investigación del Desarrollo seguro en ingeniería del software menciona que "OWASP nos provee de una serie de recursos basados, sobre todo, en guías y herramientas, para que nuestros proyectos web sean lo más seguros posibles" (p. 25). A su vez se menciona que, para desarrolladores y auditores de seguridad, proporciona numerosos recursos que facilitan la implementación del "ciclo de vida de desarrollo de software seguro (S-SDLC)", además de ofrecer una variedad de guías y herramientas para evaluar la seguridad de las aplicaciones web.

Figura 23.
Top 10 Web Application Security Risks



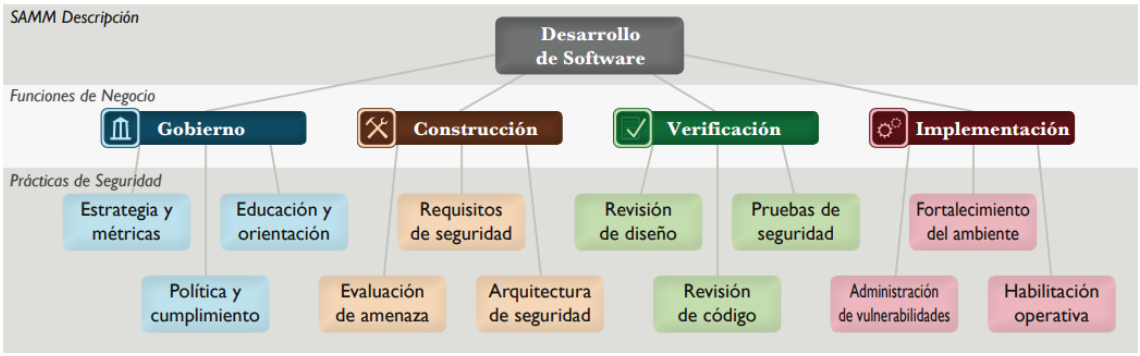
Nota. Diseño del Modelo SAMM (OWASP, 2020)

SAMM v2.0 de OWASP

El "Modelo de Madurez de Seguridad del Software" (SAMM, por sus siglas en inglés) es un marco que se enfoca en evaluar y mejorar sus prácticas de seguridad en el desarrollo de los sistemas informáticos. SAMM v2.0 ofrece una guía para comprobar la madurez de la seguridad en cinco áreas clave: primero el gobierno, continúa con el Diseño, se enfoca en la Implementación, enfatiza en la Verificación y las Operaciones como parte de la creación de software en las organizaciones como en el aprendizaje del mismo (OWASP, 2020).

> Según Pérez et al. (2023) menciona que el modelo SAMM:
>
> Incluye los estándares ISO, PCI, COBIT e ISM3. Es rápido, fácil de desplegar y ayuda a mejorar el entorno de seguridad y el estado de la construcción de software. Se caracteriza por ser personalizable y adaptable en las organizaciones, según las necesidades y los niveles de seguridad que se deseen alcanzar. Proporciona los siguientes recursos:
>
> - Evaluación de prácticas de desarrollo de software seguro.
> - Construcción de un Software seguro que sea equilibrado.
> - Mejoras en la aplicación que garantice la seguridad.
> - Definición y medición de actividades de seguridad. (p. 141)

Figura 24.
Modelo SAMM



Nota. Diseño del Modelo SAMM (Garcia, 2020).

2.2. Descripción de la propuesta

La propuesta de una guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera atiende la necesidad crítica de integrar la seguridad en todas las fases del desarrollo de software. Esta guía se basa en los principios del Ciclo de Vida de Desarrollo de Software Seguro (S-SDLC) y se adapta específicamente a las necesidades educativas del Instituto Cordillera. Al proporcionar un marco estructurado que abarca desde el modelado de amenazas hasta las pruebas de seguridad basadas en riesgos, la guía asegura que los estudiantes y educadores incorporen prácticas de seguridad en cada etapa del proceso de desarrollo.

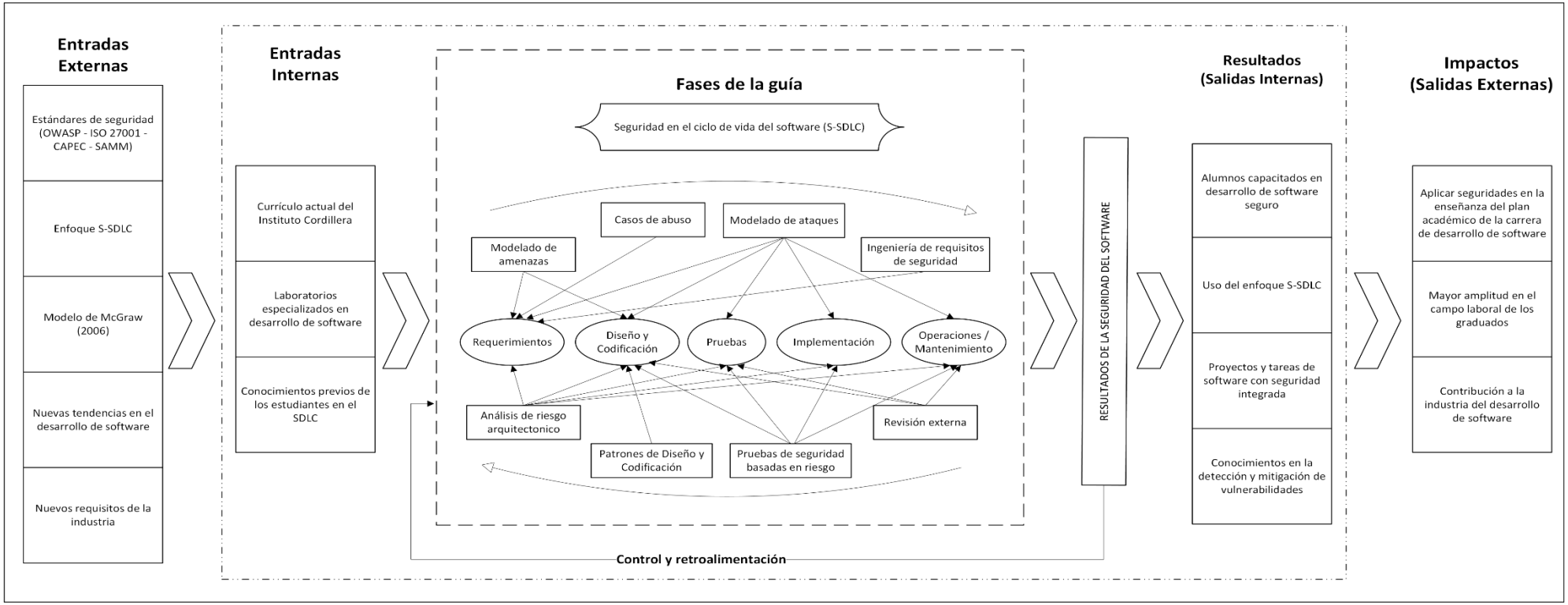
El objetivo principal de esta guía es establecer un conjunto de prácticas de seguridad que se integren de manera fluida en el currículo de desarrollo de software del Instituto Cordillera. Basándose en estándares reconocidos y mejores prácticas de la industria, como CAPEC para la enumeración y clasificación de patrones de ataque comunes, la guía proporcionará a los estudiantes las herramientas y conocimientos necesarios para desarrollar software seguro desde el inicio. Esto incluye la implementación de técnicas de modelado de amenazas, análisis de riesgos arquitectónicos, y la aplicación de patrones de diseño y codificación seguros.

La creación de esta guía no solo ayudará a mejorar la calidad de la educación en desarrollo de software en el Instituto Cordillera, sino que también preparará a los estudiantes para las demandas del mundo real en ciberseguridad. Al socializar y adaptar estas políticas y prácticas a todos los niveles del programa educativo, se creará una cultura de seguridad que los estudiantes llevarán consigo a sus futuras carreras. Además, la guía servirá como un recurso valioso para los educadores, proporcionando un marco claro para la enseñanza de prácticas de desarrollo de software seguro y ayudando a mantener el currículo actualizado con las últimas tendencias en seguridad informática

a. Estructura general

En la figura 25 se muestra el flujo de proceso de cómo se llevará a cabo la propuesta de una guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera. Este diagrama permite identificar el alcance y el desarrollo de la guía, ilustrando cómo se integran los elementos de seguridad en cada fase del ciclo de vida del desarrollo de software.

Figura 25.
Estructura general de la guía.



Nota. Organizador gráfico de la guía para la enseñanza de seguridades en el desarrollo del software.

El diagrama presenta una serie de pasos que van desde las entradas externas, como estándares de seguridad y nuevas tendencias en el desarrollo de software, pasando por las entradas internas propias del Instituto, hasta llegar a las fases centrales de la guía donde se detalla la implementación del S-SDLC. Finalmente, se muestran los resultados esperados e impactos, tanto internos como externos, que surgirán de la aplicación de esta guía. Esta estructura permite visualizar cómo la seguridad se incorpora de manera holística en el proceso educativo, alineando la enseñanza del desarrollo de software con las mejores prácticas de seguridad de la industria.

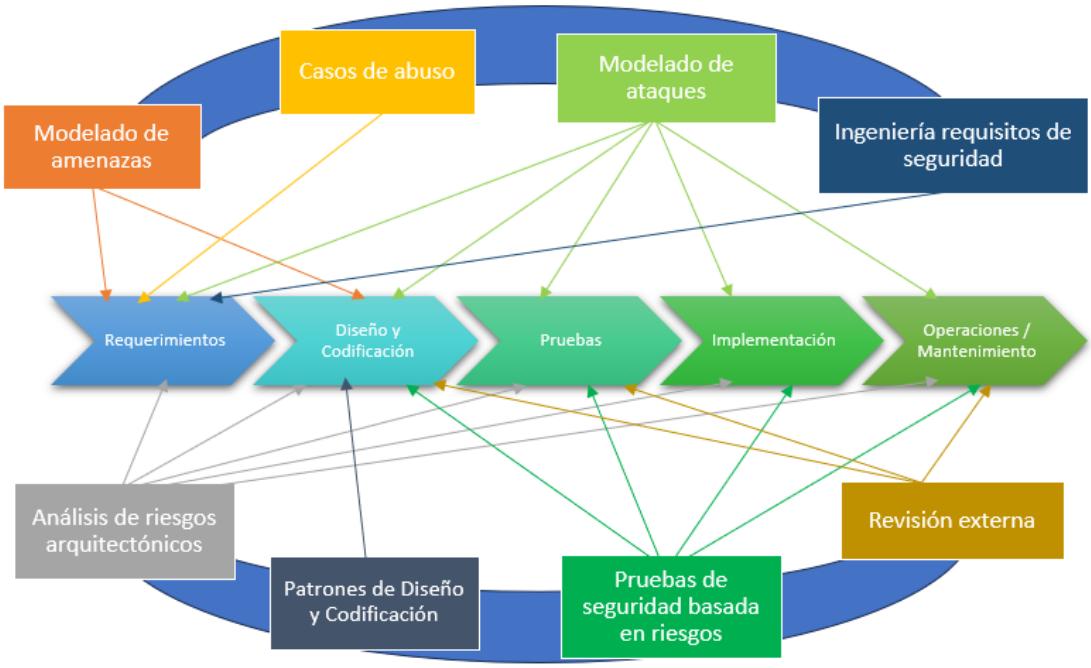
b. Explicación del aporte

La propuesta de una guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera se estructura en varios componentes interconectados, cada uno con un papel crucial en la integración de la seguridad en el proceso educativo del desarrollo de software.

Las Entradas Externas, que incluyen estándares de seguridad como OWASP e ISO 27001, el enfoque S-SDLC, el modelo de McGraw y las tendencias de la industria, proporcionan el marco de referencia y las mejores prácticas actuales. Estas se complementan con las Entradas Internas, que consideran el currículo existente del Instituto, los laboratorios especializados y los conocimientos previos de los estudiantes, asegurando que la guía se adapte al contexto específico de la institución.

El núcleo de la propuesta se encuentra en las Fases de la guía, donde se integra la seguridad en cada etapa del ciclo de vida del desarrollo de software. Esto incluye actividades como el “modelado de amenazas, casos de abuso, ingeniería de requisitos de seguridad y pruebas de seguridad basadas en riesgo”, que se entrelazan con las fases tradicionales del desarrollo de software. Este enfoque asegura que los estudiantes aprendan a considerar la seguridad como una parte integral de todo el proceso de desarrollo, no como una consideración posterior.

Figura 26.
Ciclo de vida del software seguro.



Nota. Ciclo de vida seguro del software (S-SDLC).

Los resultados esperados de implementación de esta guía incluyen alumnos capacitados en desarrollo de software seguro, proyectos con seguridad integrada y conocimiento en la detección y mitigación de vulnerabilidades. Estos resultados internos se traducen en Impactos externos significativos, como la mejora del plan académico, una mayor competitividad de los graduados en el campo laboral y una contribución sustancial a la industria del desarrollo de software seguro.

Finalmente, el componente de Control y retroalimentación asegura que la guía sea un documento vivo, capaz de evolucionar y adaptarse a los cambios en el panorama de la seguridad informática y las necesidades educativas. Este enfoque integral no solo mejora la calidad de la educación en desarrollo de software en el Instituto Cordillera, sino que también prepara a los estudiantes para enfrentar los desafíos de seguridad en el mundo real, contribuyendo así a la formación de profesionales más competentes y a la mejora general de la seguridad en el desarrollo de software.

c. Estrategias y/o técnicas

La construcción de la guía para la aplicación de seguridades se llevó a cabo mediante una serie de estrategias y técnicas cuidadosamente seleccionadas. El proceso comenzó con una investigación cuantitativa, utilizando encuestas dirigidas a los alumnos de últimos niveles y docentes de programación de la carrera de Desarrollo de Software de la institución. Esta técnica permitió evaluar el nivel actual de enseñanza en seguridades dentro del SDLC, proporcionando datos cuantificables sobre el conocimiento existente de los alumnos. El análisis estadístico de estos resultados fue crucial para identificar patrones, fortalezas y debilidades en el enfoque actual de enseñanza.

Posteriormente, se llevó a cabo una investigación documental para conceptualizar los fundamentos del S-SDLC. Esta etapa incluyó el estudio de estándares de seguridad como OWASP, ISO 27001, CAPEC, SAMM, entre otros, asegurando que la guía estuviera alineada con las mejores prácticas de la industria. Paralelamente, se llevó a cabo un análisis curricular, contrastando el plan de estudios actual del Instituto con los principios del S-SDLC, lo que permitió identificar la aplicabilidad de la guía para la integración de seguridad en el currículo existente.

El diseño de la guía siguió un enfoque iterativo, desarrollando borradores iniciales que fueron revisados y refinados basándose en la retroalimentación de especialistas en seguridad y educadores con los que cuenta la institución. Esta técnica aseguró que la guía fuera práctica, relevante y aplicable al contexto específico del Instituto. Además, se implementó una validación con especialistas en seguridad informática, lo que contribuyó a garantizar la calidad y pertinencia del producto final.

Finalmente, se realizó una prueba piloto aplicando una versión preliminar de la guía en un grupo selecto de clases. Esta estrategia permitió obtener una retroalimentación valiosa de los estudiantes sobre la efectividad y usabilidad de la guía, posibilitando realizar ajustes finales. Este enfoque multifacético y metódico en la construcción de la guía aseguró que el producto sea relevante y efectivo en la enseñanza de seguridades en el ciclo de vida del desarrollo de software en el Instituto Tecnológico Universitario Cordillera.

2.3. Validación de la propuesta

La propuesta ha sido validada por tres especialistas, cuyos detalles se pueden consultar en el ANEXO 3, quienes han proporcionado sus criterios, comentarios y recomendaciones.

Para la selección de especialistas, se ha considerado un perfil que cumple con los siguientes criterios: formación académica relacionada con el tema investigativo, experiencia académica y/o laboral en el ámbito de la informática, y motivación para participar.

La tabla a continuación ofrece información detallada sobre los actores seleccionados para la validación del modelo.

Tabla 20.
Descripción de perfil de validadores.

Nombres y Apellidos	Años de experiencia	Titulación Académica	Cargo
Jaime Neptalí Basantes Basantes	20 años	Magister Universitario en Ciberseguridad	Docente Instituto Universitario Cordillera, Director de Sistemas Informáticas FF.AA.

Jonathan Marcelo Díaz Almachi	8 años	Magister en Seguridad Informática.	Docente Instituto Universitario Cordillera, Consultor externo Ciberseguridad - Servife
Stalin Mauricio Mejía Montenegro	10 años	Ingeniero en sistemas informáticos.	Docente Instituto Universitario Cordillera, Analista de TI - Aliservis SA - Domino's Ecuador

Nota. Descripción de los especialistas.

El análisis conjunto de las tres evaluaciones de la “Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera” revela una valoración mayoritariamente positiva. Los evaluadores coinciden en que la guía es altamente pertinente, factible, y novedosa, destacando su aplicabilidad en entornos educativos y profesionales. Además, se considera que la fundamentación pedagógica es sólida y que las indicaciones para su uso son claras y eficaces. Aunque la fundamentación tecnológica recibió una evaluación ligeramente menor, sigue siendo vista como adecuada. En resumen, la guía es vista como un recurso valioso y bien diseñado para fortalecer la enseñanza de la “seguridad en el desarrollo de software”.

2.4. Matriz de articulación de la propuesta

Tabla 21.
Matriz de articulación

EJES O PARTES PRINCIPALES	SUSTENTO TEÓRICO	SUSTENTO METODOLÓGICO	ESTRATEGIAS / TÉCNICAS	DESCRIPCIÓN DE RESULTADOS	INSTRUMENTOS APLICADOS
Estudio del marco de referencia S-SDCL.	Marco de referencia para integrar la seguridad en todas las fases del ciclo de vida del desarrollo de software.	Metodología bibliográfica.	Fuente bibliográfica, enfoque integral S-SDLC.	Permite una visión integral de buenas prácticas de seguridad en el SDLC.	Investigación bibliográfica.
Diagnóstico de las prácticas actuales de la enseñanza del ciclo de vida del software en el Instituto Tecnológico Universitario Cordillera.	Proceso de investigación cuantitativa.	Encuestas.	Encuesta realizada a los alumnos y docentes de ITSCO.	Se desarrolló encuestas con el fin de diagnosticar el nivel de enseñanza de seguridades en el SDLC.	Encuestas.
Modelado de amenazas.	Lo define como una herramienta válida para evaluar los riesgos inherentes a una aplicación durante su desarrollo (Bermejo, 2019).	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Permite identificar y mitigar posibles amenazas durante el desarrollo del software, asegurando que los riesgos sean abordados de manera efectiva desde las fases iniciales del SDLC.	Fuente bibliográfica.

Casos de abuso.	Un caso de abuso es la inversa de un caso de uso, es decir, una función que el sistema no debe permitir o una secuencia completa de acciones que resulta en una pérdida para la organización (Xiaohong et al., 2015).	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Ayuda a los estudiantes a entender cómo evitar comportamientos maliciosos en el software, enseñándoles a identificar posibles abusos de las funcionalidades y a desarrollar contramedidas para prevenirlos.	Fuente bibliográfica.
Modelado de ataques.	Mecanismo o medio para capturar y representar la perspectiva y el conocimiento del ciberatacante con el suficiente detalle acerca de cómo los ataques llevan a cabo los métodos más frecuentes de explotación (exploit) y las técnicas usadas para comprometer el software (Bermejo, 2019).	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Facilita la comprensión profunda de los métodos de ataque y explotación, proporcionando a los estudiantes las herramientas necesarias para anticipar y contrarrestar posibles amenazas, integrando estas prácticas en los procesos de diseño y desarrollo del software en el SDLC.	Fuente bibliográfica.
Ingeniería requisitos de seguridad.	La etapa de ingeniería de requisitos abarca todas las actividades y tareas necesarias antes de comenzar el diseño, con el objetivo principal de definir los requisitos que determinan los aspectos funcionales y no	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Asegura que los estudiantes comprendan la importancia de definir y especificar requisitos de seguridad desde el inicio del proyecto, garantizando que los productos de software cumplan con los estándares	Fuente bibliográfica.

	funcionales del software (Retena, 2023).			de seguridad necesarios, lo que se traduce en aplicaciones más seguras y confiables.	
Análisis de riesgos arquitectónicos.	El proceso de análisis de riesgo arquitectónico que desarrolla un acercamiento a un análisis de riesgo que implica tres pasos básicos: análisis de resistencia al ataque a la ambigüedad y a la debilidad (Hazlegreaves, 2021).	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Permite evaluar la arquitectura del software en busca de posibles vulnerabilidades, mejorando su capacidad para diseñar sistemas que sean robustos y resilientes frente a ataques.	Fuente bibliográfica.
Patrones de Diseño y Codificación.	Los patrones de diseño se definen como soluciones generales y repetibles para problemas recurrentes en la ingeniería de software, diseñados específicamente para contribuir a la creación de software que sea menos vulnerable, más resistente, y capaz de tolerar ataques. (McGraw, 2006).	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Proporciona un conjunto de buenas prácticas y soluciones repetibles para el diseño de software seguro, permitiéndoles desarrollar aplicaciones que sean intrínsecamente más seguras y resistentes a ataques, fortaleciendo así la enseñanza de seguridades en el “ciclo de vida del software” en el entorno académico de la Institución.	Fuente bibliográfica.
Pruebas de seguridad basada en riesgos.	Al identificar los riesgos del sistema y diseñar las pruebas desde la perspectiva de un atacante, un	Metodología bibliográfica.	Fuente bibliográfica, marco de	Otorga las habilidades para diseñar y realizar pruebas de seguridad focalizadas en las áreas más	Fuente bibliográfica.

	probador de seguridad de software puede centrarse eficazmente en las áreas del código donde un ataque podría tener mayor probabilidad de éxito (Bermejo, 2019).		referencia S-SDLC.	vulnerables del código, incrementando la efectividad de las pruebas y asegurando que los productos de software desarrollados estén mejor protegidos contra posibles ataques.	
Revisión externa.	Un análisis externo por personal ajeno al equipo de diseño es bastante eficaz y fundamental aportando otra visión de la seguridad del sistema y del riesgo y contribuyendo a una mejora de la seguridad, porque seguramente este análisis va a descubrir alguna amenaza y riesgo residual existente (Bermejo, 2019).	Metodología bibliográfica.	Fuente bibliográfica, marco de referencia S-SDLC.	Introduce a los estudiantes en la práctica de la revisión externa, permitiéndoles comprender la importancia de una perspectiva externa para identificar amenazas y riesgos residuales que podrían no haber sido detectados internamente.	Fuente bibliográfica.

Nota. Detalle de la matriz de articulación.

CONCLUSIONES

La elaboración de una guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los principios del S-SDLC demuestra ser una herramienta fundamental para fortalecer la formación académica de los estudiantes de la carrera de desarrollo de software en el Instituto Tecnológico Universitario Cordillera. Esta guía no solo refuerza la comprensión teórica de los conceptos de seguridad, sino que también ayuda en la aplicación de prácticas y técnicas para la enseñanza del desarrollo seguro, promoviendo la creación de software de alta calidad y resiliente frente a amenazas.

La contextualización de los fundamentos teóricos sobre el desarrollo de software seguro, basado en los principios del S-SDLC, ha permitido establecer una base sólida para la enseñanza de estos conceptos en la carrera de desarrollo de software. Esto asegura que los estudiantes comprendan no sólo las técnicas de programación segura, sino también la importancia de integrar la seguridad desde las primeras etapas del ciclo de vida del software fundamentados ya en un marco integral que se usa a nivel empresarial en la actualidad.

El diagnóstico de las prácticas actuales en la enseñanza del ciclo de vida del software en el Instituto Tecnológico Universitario Cordillera ha revelado áreas clave de mejora en cuanto a la incorporación de seguridad en el currículo. Los resultados de las encuestas a docentes y estudiantes proporcionan un panorama claro de las fortalezas y debilidades actuales, lo que permitirá orientar mejor la enseñanza hacia la formación de profesionales capaces de desarrollar software seguro desde su concepción.

El diseño de una guía para la aplicación de técnicas de desarrollo de software seguro, fundamentada en el marco de referencia S-SDLC, representa un paso crucial para institucionalizar estas prácticas dentro del currículo académico. Esta guía servirá como un recurso didáctico esencial que ayudará a la integración de la seguridad en las actividades de desarrollo, contribuyendo a la formación de estudiantes más preparados para enfrentar los desafíos del desarrollo de software en el entorno actual.

RECOMENDACIONES

Se recomienda implementar un sistema integral de revisión y actualización anual de la guía. Este proceso debe incluir la formación de un comité de revisión compuesto por docentes y profesionales de la industria. Se sugiere realizar talleres semestrales para recopilar feedback sobre la aplicación práctica de la guía, analizar las últimas tendencias en seguridad de software y adaptar el contenido según sea necesario. Es crucial publicar actualizaciones anuales con cambios documentados y justificados, asegurando así la relevancia continua de la guía en el cambiante panorama de la seguridad informática.

El incorporar una plataforma en línea con módulos interactivos de aprendizaje, una biblioteca digital de casos de estudio reales categorizados por tipo de vulnerabilidad y sector de la industria, y webinars mensuales con expertos en seguridad de software, el implementar foros de discusión moderados por docentes para facilitar el intercambio de ideas y la resolución de dudas entre los estudiantes, fomentando así un aprendizaje colaborativo y actualizado.

Incluir encuestas semestrales a estudiantes y docentes sobre la efectividad de la enseñanza de seguridad, así como un análisis comparativo de los proyectos estudiantiles antes y después de la implementación de la guía a su vez organizar sesiones de retroalimentación con grupos focales de estudiantes de diferentes niveles y realizar evaluaciones prácticas para medir la aplicación de conceptos de seguridad en proyectos reales, proporcionando así una visión completa del impacto de la guía en el aprendizaje.

Se sugiere crear un comité de diseño y adaptación que se reúna trimestralmente para revisar el contenido y la estructura del contenido del plan académico en base a las actuales seguridades en el software. Este comité debería colaborar con empresas de desarrollo de software para incorporar escenarios reales, incorporando un conjunto de métricas para evaluar la efectividad de cada sección de la guía, y crear versiones adaptadas para diferentes niveles de estudio. Este enfoque asegurará que la guía se mantenga práctica, relevante y accesible para todos los estudiantes, independientemente de su nivel de experiencia.

BIBLIOGRAFÍA

Álvarez, G. (24 de septiembre de 2018). *Kyocode*. Ciclo de vida de un software: https://www.kyocode.com/2018/09/ciclo-de-vida-de-un-software/

Bermejo, J. (2019). Desarrollo Seguro de Software y Auditoría de la Ciberseguridad. *UNIR - Electronics, 8*(11), 1218. https://doi.org/https://doi.org/10.3390/electronics8111218

Díaz, G. (2015). *Estudio de Técnicas Automáticas de Análisis de Vulnerabilidades de Seguridad en Aplicaciones Web.* UNED.

Garcia, V. (2020). Metodologías de desarrollo de software seguro con propiedades agiles. *Universidad Laica Eloy Alfaro de Manabí (ULEAM), 5*(10), 1027-1046. https://doi.org/10.23857/pc.v5i10.2658

Hazlegreaves, S. (15 de octubre de 2021). *Education sector suffers series of cyber attacks in 2021. Open Access Government.* https://www.openaccessgovernment.org/education-sector-suffers-series-of-cyber-attacks-in-2021/122465/

Hernández, E. O. (2023). *Modelado de Amenazas de una Aplicación Seguridad en el software.* Universidad UNIR.

Howell, J. (25 de Agosto de 2022). *Threat Modeling Tool feature overview*. https://learn.microsoft.com/es-es/azure/security/develop/threat-modeling-tool-feature-overview

IBM. (10 de mayo de 2024). *IBM*. ¿Qué es el desarrollo de software?: https://www.ibm.com/es-es/topics/software-development

López, Á. D. (2020). Método para el desarrollo de software seguro basado en la ingeniería de software y ciberseguridad. *Universidad ECOTEC, Ecuador, 5*(3.1), 263-280. https://doi.org/10.33890/innova.v5.n3.1.2020.1440

McGraw, G. (2006). ResearchGate. *Berryville Institute of Machine Learning, 17*(1), 7-10. https://doi.org/10.1109/ISSRE.2006.43

Microsoft. (3 de febrero de 2010). *Microsoft*. Simplified Implementation of the Microsoft SDL: https://www.microsoft.com/en-us/download/details.aspx?id=12379

Ortega, J. (2020). Desarrollo seguro en ingeniería del software: Aplicaciones seguras con Android, NodeJS, Python y C++. Alpha Editorial. https://books.google.es/books?id=x3J6EAAAQBAJ&lpg=PA7&hl=es&pg=PA4#v=onepage&q&f=false

OWASP. (19 de enero de 2020). *OWASP Top Ten*. OWASP Foundation: https://owasp.org/www-project-top-ten/#

Pachacuti, M. (2020). *MODELO DE SEGURIDAD PARA EL DESARROLLO DE SOFTWARE* [Tesis de maestria, UNIVERSIDAD MAYOR DE SAN ANDRÉS]. Repositorio institucional. https://repositorio.umsa.bo/xmlui/bitstream/handle/123456789/27697/TM-3721.pdf?sequence=1&isAllowed=y

Parraga, J. (marzo de 2024). *Propuesta de plan de seguridad para mitigar vulnerabilidades en el ciclo de integración continua y despliegue de aplicaciones web en DevOps.* http://repositorio.uisrael.edu.ec/handle/47000/4135

Pérez, O., Gainza, D., & Raúl, H. (29 de enero de 2023). *Estudio del desarrollo seguro del software*. https://publicaciones.uci.cu/index.php/serie/article/view/1291

Retena, P. (Junio de 2023). *Análisis y Modelado de Amenazas* . https://prezi.com/uky4fywbiexu/analisis-y-modelado-de-amenazas/

Sampieri, R. H. (2014). Metodología de la investigación. méxico: mcGraw-Hill. https://apiperiodico.jalisco.gob.mx/api/sites/periodicooficial.jalisco.gob.mx/files/metodologia_de_la_investigacion_-_roberto_hernandez_sampieri.pdf

Sierra, T. (2022). *LA SEGURIDAD INFORMÁTICA EN EL DESARROLLO DE APLICACIONES WEB* [UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD]. Respositorio institucional. https://repository.unad.edu.co/bitstream/handle/10596/54049/ta52sie605.pdf?sequence=1&isAllowed=y

Trujillo, D., y Chávez, A. (2016). *Sistema de Control de Versiones para el Desarrollo de Software Seguro* [Trabajo investigativo, Fundación Universitaria los Libertadores]. Repositorio institucional. https://repository.libertadores.edu.co/server/api/core/bitstreams/416eb844-5e4b-4a06-a44d-c8d65dc654bb/content

Viteri, C. (marzo de 2023). *Propuesta de Políticas de seguridad informática en el desarrollo de sistemas web, aplicando la norma ISO 27002, para la Universidad Central del Ecuador.* http://repositorio.uisrael.edu.ec/handle/47000/4141

Xiaohong, Y., Emmanuel, N., Imano, W., y Huiming, Y. (2015). Developing Abuse Cases Based on Threat Modeling and Attack Patterns. Journal of Software. https://doi.org/10.17706/jsw.10.4.491-498.

ANEXOS

ANEXO 1

FORMATO DE ENCUESTA PARA LOS ALUMNOS

Pregunta 1. Durante su formación académica, ¿se le ha mencionado sobre el concepto de "Ciclo de Vida del Desarrollo de Software Seguro (S-SDLC)"?

Si

No

Pregunta 2. Durante su formación académica, ¿se le ha instruido sobre cómo aplicar revisiones de código estáticas y dinámicas, el uso de entornos aislados (Sandbox), y pruebas de penetración (pentesting)?

Si

No

Pregunta 3. ¿En el aprendizaje de su carrera ha cumplido con todos los procesos del Ciclo de Vida de Desarrollo de Software (SDLC) tales cómo?

Requerimientos	**Si**	**No**
Diseño - Codificación	**Si**	**No**
Pruebas	**Si**	**No**
Implementación (publicar el sistema en un servidor local, público)	**Si**	**No**
Mantenimiento	**Si**	**No**
Prácticas de seguridad en cada una de las fases del ciclo de vida del software	**Si**	**No**

Pregunta 4. En tu opinión, ¿Conoces si en los planes analíticos (PA) de las materias de programación incluyen alguna metodología para el desarrollo de software seguro?

Si

No

Pregunta 5. ¿Utilizas prácticas de codificación segura en tus proyectos o deberes de las materias de programación?

Siempre

A veces

Rara vez

Nunca

Pregunta 6. ¿Has participado en alguna revisión de código enfocada en la seguridad en algún nivel de la carrera?

Si

No

Pregunta 7. ¿Te sientes preparado para aplicar medidas de seguridad en tus proyectos de desarrollo de software fuera del ámbito académico?

Si

No

Pregunta 8. ¿Consideras que el plan de estudios actual de la carrera incluye suficientes contenidos sobre seguridad en el desarrollo de software?

Si

No

Pregunta 9. ¿Crees que la seguridad debería ser un tema transversal en todos los cursos relacionados con el desarrollo de software?

Si

No

Pregunta 10. ¿Considera que el aprendizaje de prácticas de seguridad en el desarrollo de software es importante para su carrera profesional?

Si

No

ANEXO 2

FORMATO DE ENCUESTA PARA LOS DOCENTES

Pregunta 1. ¿Incluye temas de seguridad en el desarrollo de software en sus asignaturas de programación?

Si

No

Pregunta 2. ¿Utiliza herramientas de análisis de seguridad de código como OWASP ZAP, Syhunt, Wireshark, Fortify, SonarQube, etc., en sus clases de programación?

Si

No

Pregunta 3. ¿Con qué frecuencia discute sobre seguridad en el desarrollo de software en sus clases?

Siempre

Frecuentemente

Ocasionalmente

Rara vez

Nunca

Pregunta 4. ¿En la enseñanza de su materia de programación cree usted que ha cumplido con todos los procesos del Ciclo de Vida de Desarrollo de Software (SDLC) cómo?

Análisis de requisitos (Requerimientos)	**Si**	**No**
Diseño - Codificación	**Si**	**No**
Implementación (publicar el sistema en un servidor local, público)	**Si**	**No**
Pruebas	**Si**	**No**
Mantenimiento	**Si**	**No**
Prácticas de seguridad en cada una de las fases del ciclo de vida del software	**Si**	**No**

Pregunta 5. ¿Cree que la seguridad es adecuadamente abordada en el actual plan de estudios de la carrera de Desarrollo de Software?

Si

No

Pregunta 6. ¿Considera que sus estudiantes están adquiriendo suficientes habilidades para desarrollar software seguro?

Si

No

Pregunta 7. ¿Cree que debería haber más enfoque en la seguridad en la formación de los docentes de programación?

Si

No

Pregunta 8. ¿Estaría interesado en recibir formación adicional o participar en talleres sobre desarrollo de software seguro?

Si

No

ANEXO 3

VALIDACIÓN DE ESPECIALISTAS

UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMÁTICA

INSTRUMENTO PARA VALIDACIÓN DE LA PROPUESTA

Estimado colega:

Se solicita su valiosa cooperación para evaluar la calidad del siguiente contenido digital **"Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera"**. Sus criterios son de suma importancia para la realización de este trabajo, por lo que se le pide que brinde su cooperación contestando las preguntas que se realizan a continuación.

Datos informativos

Validado por: Jaime Neptalí Basantes Basantes
Título obtenido: Magister Universitario en Ciberseguridad
C.I.: 1706366794
E-mail: jaime.basantes@cordillera.edu.ec
Institución de Trabajo: Instituto Universitario Cordillera, FF.AA.
Cargo: Docente, Director de Sistemas Informáticas
Años de experiencia en el área: 20 años

Instructivo:

- Responda cada criterio con la máxima sinceridad del caso.
- Revisar, observar y analizar la propuesta de la plataforma virtual, blog o sitio web.
- Coloque una X en cada indicador, tomando en cuenta que Muy adecuado equivale a 5, Bastante Adecuado equivale a 4, Adecuado equivale a 3, Poco Adecuado equivale a 2 e Inadecuado equivale a 1.

Tema: "Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera"

Indicadores	Muy adecuado	Bastante Adecuado	Adecuado	Poco adecuado	Inadecuado
Pertinencia	X				
Aplicabilidad	X				
Factibilidad	X				
Novedad	X				
Fundamentación pedagógica	X				
Fundamentación tecnológica		X			
Indicaciones para su uso	X				
TOTAL	30	4			

Observaciones: La estructura bien organizada y el contenido actualizado de la guía desde mi percepción demuestran una comprensión profunda de las necesidades actuales de la industria del software en materia de seguridad. La inclusión de casos de estudio reales y ejemplos prácticos permite a los estudiantes relacionar los conceptos teóricos con situaciones de nuestra época actual.

Recomendaciones: Para maximizar el impacto de la guía, se recomienda implementar un programa de capacitación para los docentes del Instituto. Este programa debería no solo familiarizarlos con el contenido de la guía, sino también proporcionarles estrategias pedagógicas para enseñar conceptos de seguridad en el desarrollo de software de manera efectiva.

Lugar, fecha de validación: Quito D.M. 29 de octubre de 2024

Universidad Israel

ESPOG | Escuela de Posgrados

AUTORIZACIÓN PARA EL TRATAMIENTO DE DATOS PERSONALES

La Universidad Tecnológica Israel con domicilio en Francisco Pizarro E4-142 y Marieta de Veintimilla, Quito – Ecuador y dirección electrónica de contacto protecciondatospersonales@uisrael.edu.ec es la entidad responsable del tratamiento de sus datos personales, cumple con informar que la gestión de sus datos personales es con la finalidad de registrar el instrumento de validación de propuesta de la Maestría en Seguridad Informática, como requisito de titulación para los cursantes del programa de posgrados. Como consecuencia de este tratamiento sus datos estarán públicos en el repositorio donde reposan los trabajos de titulación.

La base legal para realizar dicho tratamiento es su consentimiento otorgado en este documento, el mismo que puede revocarlo en cualquier momento.

Los datos personales se publicarán en el repositorio de trabajos de titulación, no se comunicarán a terceros con otra finalidad distinta a la recogida, salvo cuando exista una obligación legal, orden judicial, de agencia o entidad gubernamental con facultades comprobadas, o de autoridad competente.

En algunos casos este tratamiento puede implicar transferencias internacionales de datos, para lo cual garantizamos el cumplimiento de la Ley Orgánica de Protección de Datos Personales y el Reglamento a la ley. La UISRAEL conservará sus datos durante el tiempo necesario para que se cumpla la finalidad indicada, mientras se mantenga la relación comercial o contractual, Ud. no revoque su consentimiento o durante el tiempo necesario que resulten de aplicación por plazos legales de prescripción.

La UISRAEL ha adoptado diversas medidas organizativas, legales y tecnológicas para proteger sus datos personales. Estas medidas están diseñadas para garantizar un nivel razonable de seguridad y cumplir con las exigencias conforme a la normativa aplicable en materia de protección de datos personales.

La UISRAEL le informa que tiene derechos sobre sus datos personales conforme lo establecido en la Ley Orgánica de Protección de Datos Personales, para su ejercicio puede hacerlo mediante envío de una solicitud al correo protecciondatospersonales@uisrael.edu.ec.

Para obtener más detalles de cómo se manejan sus datos personales, la UISRAEL pone a su disposición la política de Privacidad y Protección de Datos Personales disponible en el siguiente link: Política de Protección de Datos Personales | UISRAEL

Por lo expuesto, declaro haber sido informado sobre el tratamiento de los datos personales que he entregado a la UISRAEL.

Firma del especialista
Jaime Neptalí Basantes Basantes

UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMÁTICA

INSTRUMENTO PARA VALIDACIÓN DE LA PROPUESTA

Estimado colega:

Se solicita su valiosa cooperación para evaluar la calidad del siguiente contenido digital **"**Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera**"**. Sus criterios son de suma importancia para la realización de este trabajo, por lo que se le pide que brinde su cooperación contestando las preguntas que se realizan a continuación.

Datos informativos

Validado por: Jonathan Marcelo Díaz Almachi
Título obtenido: Magister en Seguridad Informática
C.I.: 1722467923
E-mail: jonathan.diaz@cordillera.edu.ec
Institución de Trabajo: Instituto Universitario Cordillera, Servife
Cargo: Docente, Consultor externo Ciberseguridad
Años de experiencia en el área: 8 años

Instructivo:

- Responda cada criterio con la máxima sinceridad del caso.
- Revisar, observar y analizar la propuesta de la plataforma virtual, blog o sitio web.
- Coloque una X en cada indicador, tomando en cuenta que Muy adecuado equivale a 5, Bastante Adecuado equivale a 4, Adecuado equivale a 3, Poco Adecuado equivale a 2 e Inadecuado equivale a 1.

Tema: "Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera"

Indicadores	Muy adecuado	Bastante Adecuado	Adecuado	Poco adecuado	Inadecuado
Pertinencia		X			
Aplicabilidad	X				
Factibilidad		X			
Novedad	X				
Fundamentación pedagógica	X				
Fundamentación tecnológica	X				
Indicaciones para su uso	X				
TOTAL	25	8			

Observaciones: La flexibilidad de la guía para adaptarse a diferentes niveles de conocimiento y su énfasis en la mejora continua son aspectos particularmente destacables. Esto no solo facilita su implementación en diversos cursos, sino que también asegura su relevancia a largo plazo en un campo tan dinámico como la seguridad informática.

Recomendaciones: Se recomienda crear un repositorio digital interactivo que complemente la guía física. Este repositorio podría incluir videos explicativos, ejercicios prácticos actualizados regularmente, foros de discusión moderados y enlaces a recursos externos relevantes.

Lugar, fecha de validación: Quito D.M. 29 de octubre de 2024

Universidad Israel

ESPOG | Escuela de Posgrados

AUTORIZACIÓN PARA EL TRATAMIENTO DE DATOS PERSONALES

La Universidad Tecnológica Israel con domicilio en Francisco Pizarro E4-142 y Marieta de Veintimilla, Quito – Ecuador y dirección electrónica de contacto protecciondatospersonales@uisrael.edu.ec es la entidad responsable del tratamiento de sus datos personales, cumple con informar que la gestión de sus datos personales es con la finalidad de registrar el instrumento de validación de propuesta de la Maestría en Seguridad Informática, como requisito de titulación para los cursantes del programa de posgrados. Como consecuencia de este tratamiento sus datos estarán públicos en el repositorio donde reposan los trabajos de titulación.

La base legal para realizar dicho tratamiento es su consentimiento otorgado en este documento, el mismo que puede revocarlo en cualquier momento.

Los datos personales se publicarán en el repositorio de trabajos de titulación, no se comunicarán a terceros con otra finalidad distinta a la recogida, salvo cuando exista una obligación legal, orden judicial, de agencia o entidad gubernamental con facultades comprobadas, o de autoridad competente.

En algunos casos este tratamiento puede implicar transferencias internacionales de datos, para lo cual garantizamos el cumplimiento de la Ley Orgánica de Protección de Datos Personales y el Reglamento a la ley. La UISRAEL conservará sus datos durante el tiempo necesario para que se cumpla la finalidad indicada, mientras se mantenga la relación comercial o contractual, Ud. no revoque su consentimiento o durante el tiempo necesario que resulten de aplicación por plazos legales de prescripción.

La UISRAEL ha adoptado diversas medidas organizativas, legales y tecnológicas para proteger sus datos personales. Estas medidas están diseñadas para garantizar un nivel razonable de seguridad y cumplir con las exigencias conforme a la normativa aplicable en materia de protección de datos personales.

La UISRAEL le informa que tiene derechos sobre sus datos personales conforme lo establecido en la Ley Orgánica de Protección de Datos Personales, para su ejercicio puede hacerlo mediante envío de una solicitud al correo protecciondatospersonales@uisrael.edu.ec.

Para obtener más detalles de cómo se manejan sus datos personales, la UISRAEL pone a su disposición la política de Privacidad y Protección de Datos Personales disponible en el siguiente link: Política de Protección de Datos Personales | UISRAEL

Por lo expuesto, declaro haber sido informado sobre el tratamiento de los datos personales que he entregado a la UISRAEL.

Firma del especialista
Jonathan Marcelo Díaz Almachi

UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMÁTICA

INSTRUMENTO PARA VALIDACIÓN DE LA PROPUESTA

Estimado colega:

Se solicita su valiosa cooperación para evaluar la calidad del siguiente contenido digital **"**Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera**"**. Sus criterios son de suma importancia para la realización de este trabajo, por lo que se le pide que brinde su cooperación contestando las preguntas que se realizan a continuación.

Datos informativos

Validado por: Stalin Mauricio Mejía Montenegro
Título obtenido: Ingeniero en sistemas informáticos
C.I.: 1725667065
E-mail: Stalin.mejia@cordillera.edu.ec
Institución de Trabajo: Instituto Universitario Cordillera, Aliservis SA - Domino's Ecuador
Cargo: Docente, Analista de TI
Años de experiencia en el área: 10 años

Instructivo:

- Responda cada criterio con la máxima sinceridad del caso.
- Revisar, observar y analizar la propuesta de la plataforma virtual, blog o sitio web.
- Coloque una X en cada indicador, tomando en cuenta que Muy adecuado equivale a 5, Bastante Adecuado equivale a 4, Adecuado equivale a 3, Poco Adecuado equivale a 2 e Inadecuado equivale a 1.

Tema: "Guía para la aplicación de seguridades en la enseñanza del Ciclo de Vida del Software basada en los fundamentos del S-SDLC para el Instituto Tecnológico Universitario Cordillera"

Indicadores	Muy adecuado	Bastante Adecuado	Adecuado	Poco adecuado	Inadecuado
Pertinencia	X				
Aplicabilidad	X				
Factibilidad	X				
Novedad	X				
Fundamentación pedagógica	X				
Fundamentación tecnológica		X			
Indicaciones para su uso	X				
TOTAL	30	4			

Observaciones: La guía para la aplicación de seguridades en la enseñanza representa un avance significativo en la formación de los estudiantes de la carrera de desarrollo de software del Instituto. Su enfoque integral, que abarca desde los fundamentos teóricos hasta la aplicación práctica, proporciona a los estudiantes una base sólida para desarrollar software seguro desde las etapas iniciales del proceso de desarrollo.

Recomendaciones: Para maximizar el impacto de la guía, se recomienda implementar un programa de capacitación integral para los docentes del Instituto. Este programa debería no solo familiarizar a los educadores con el contenido de la guía, sino también proporcionarles estrategias pedagógicas efectivas para enseñar conceptos de seguridad en el desarrollo de software.

Lugar, fecha de validación: Quito D.M. 29 de octubre de 2024

Universidad Israel

ESPOG | Escuela de Posgrados

AUTORIZACIÓN PARA EL TRATAMIENTO DE DATOS PERSONALES

La Universidad Tecnológica Israel con domicilio en Francisco Pizarro E4-142 y Marieta de Veintimilla, Quito – Ecuador y dirección electrónica de contacto protecciondatospersonales@uisrael.edu.ec es la entidad responsable del tratamiento de sus datos personales, cumple con informar que la gestión de sus datos personales es con la finalidad de registrar el instrumento de validación de propuesta de la Maestría en Seguridad Informática, como requisito de titulación para los cursantes del programa de posgrados. Como consecuencia de este tratamiento sus datos estarán públicos en el repositorio donde reposan los trabajos de titulación.

La base legal para realizar dicho tratamiento es su consentimiento otorgado en este documento, el mismo que puede revocarlo en cualquier momento.

Los datos personales se publicarán en el repositorio de trabajos de titulación, no se comunicarán a terceros con otra finalidad distinta a la recogida, salvo cuando exista una obligación legal, orden judicial, de agencia o entidad gubernamental con facultades comprobadas, o de autoridad competente.

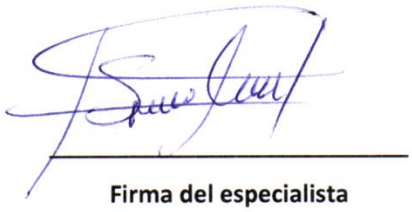
En algunos casos este tratamiento puede implicar transferencias internacionales de datos, para lo cual garantizamos el cumplimiento de la Ley Orgánica de Protección de Datos Personales y el Reglamento a la ley. La UISRAEL conservará sus datos durante el tiempo necesario para que se cumpla la finalidad indicada, mientras se mantenga la relación comercial o contractual, Ud. no revoque su consentimiento o durante el tiempo necesario que resulten de aplicación por plazos legales de prescripción.

La UISRAEL ha adoptado diversas medidas organizativas, legales y tecnológicas para proteger sus datos personales. Estas medidas están diseñadas para garantizar un nivel razonable de seguridad y cumplir con las exigencias conforme a la normativa aplicable en materia de protección de datos personales.

La UISRAEL le informa que tiene derechos sobre sus datos personales conforme lo establecido en la Ley Orgánica de Protección de Datos Personales, para su ejercicio puede hacerlo mediante envío de una solicitud al correo protecciondatospersonales@uisrael.edu.ec.

Para obtener más detalles de cómo se manejan sus datos personales, la UISRAEL pone a su disposición la política de Privacidad y Protección de Datos Personales disponible en el siguiente link: Política de Protección de Datos Personales | UISRAEL

Por lo expuesto, declaro haber sido informado sobre el tratamiento de los datos personales que he entregado a la UISRAEL.



Firma del especialista
Stalin Mauricio Mejía Montenegro

ANEXO 4

GUÍA PARA LA APLICACIÓN DE SEGURIDADES EN LA ENSEÑANZA DEL DESARROLLO DEL SOFTWARE

(S-SDLC)

GUÍA PARA LA APLICACIÓN DE SEGURIDADES EN LA ENSEÑANZA DEL DESARROLLO DEL SOFTWARE

(S-SDLC)

2024

GUÍA PARA LA APLICACIÓN DE SEGURIDADES EN LA ENSEÑANZA DEL DESARROLLO DEL SOFTWARE (S-SDLC)

2024

GUÍA PARA LA APLICACIÓN DE SEGURIDADES EN LA ENSEÑANZA DEL DESARROLLO DEL SOFTWARE (S-SDLC)

ELABORADOR POR:	Lino Jesús Cajas Pacheco - **DOCENTE DE LA CARRERA DE DESARROLLO DE SOFTWARE**

CONTENIDO

Índice de tablas

Índice de figuras

DATOS INFORMATIVOS DE LA GUÍA

Carrera	: Desarrollo de Software
Nivel	: Primero - Cuarto
Modalidad	: Presencial
Jornada	: Matutina - Nocturna
Asignatura	: Ingeniería de requerimientos, Análisis y diseño de software, Asignaturas de Programación
Campo de formación	: 48 Informática
Periodo académico	: 2024
Docente/s	: Lino Cajas, Ingeniero en Sistemas Informáticos, Stalin Mejía, Ingeniero en Sistemas Informáticos, Leonel Peñarrieta, Ingeniero en Sistemas Informáticos, Jonathan Diaz, Magister en Seguridad Informática, Jaime Basantes, Magister Universitario en Ciberseguridad,
Dirección/es electrónica/s	: lino.cajas@cordillera.edu.ec, stalin.mejia@codillera.edu.ec, leonel.peñarrieta@codillera.edu.ec, jonathan.diaz@codillera.edu.ec, jaime.basantes@codillera.edu.ec,

FUNDAMENTACIÓN DE LA GUÍA

En la actualidad, la seguridad en el desarrollo de software se ha convertido en un aspecto esencial para garantizar la confiabilidad y protección de las aplicaciones que utilizamos diariamente. La implementación de un Ciclo de Vida del Desarrollo de Software Seguro (S-SDLC) es clave en este contexto, ya que asegura que la seguridad esté integrada en cada fase del proceso de desarrollo, desde la planificación y diseño hasta la implementación y mantenimiento.

Para los estudiantes, comprender y aplicar el S-SDLC no solo es fundamental para crear software seguro y resistente frente a amenazas, sino que también les proporciona habilidades indispensables para su futuro profesional en un entorno donde la seguridad informática es cada vez más vital. La práctica de estos conocimientos mediante proyectos que involucren revisiones de código, pruebas de penetración, y la implementación de controles de seguridad durante todo el ciclo de vida del software, prepara a los estudiantes para enfrentar los desafíos actuales de ciberseguridad.

Los estudiantes pueden aplicar estos conceptos a través de una variedad de proyectos prácticos, tales como el desarrollo de aplicaciones seguras, la realización de auditorías de seguridad, y la implementación de pruebas de vulnerabilidades. Estas actividades no solo refuerzan su

comprensión teórica, sino que también les permiten adquirir experiencia práctica, imprescindible en un campo en constante evolución y de creciente importancia en la industria tecnológica.

OBJETIVOS

- Desarrollar actividades prácticas que refuercen la comprensión de los conceptos y procedimientos necesarios para la creación de software seguro descritos en esta guía, fortaleciendo así las habilidades técnicas y analíticas en el ámbito de la ciberseguridad, con el fin de mejorar sus habilidades en el saber hacer.
- Alcanzar los siguientes Resultados de Aprendizaje:
  - ✓ *Comprender los principios fundamentales del Ciclo de Vida del Desarrollo de Software Seguro (S-SDLC) en la creación de aplicaciones, asegurando que cada etapa del desarrollo contemple la seguridad desde el diseño hasta la implementación y mantenimiento.*
  - ✓ *Aplicar métodos y técnicas de seguridad informática, como revisiones de código y pruebas de penetración, para identificar y mitigar posibles vulnerabilidades en el software.*
  - ✓ *Desarrollar controles de seguridad eficaces en las arquitecturas de software, utilizando tecnologías adecuadas y ajustadas a las necesidades de seguridad de cada proyecto.*

PREPARACIÓN PREVIA DEL ESTUDIANTE

- Los estudiantes deben revisar el contenido relacionado con el Ciclo de Vida del Desarrollo de Software Seguro (S-SDLC) antes de las clases, para comprender los principios fundamentales de seguridad que se abordarán en la jornada de trabajo, así también, los resultados de aprendizaje que debe alcanzar.
- Es recomendable que el estudiante repase ejercicios y casos prácticos previos relacionados con la seguridad informática y el desarrollo seguro de software, para entender los métodos y estrategias de mitigación de riesgos que han funcionado en casos de estudios anteriores.
- Los estudiantes pueden apoyarse en la bibliografía (básica y de consulta) para ampliar su comprensión de los temas sobre seguridad en el desarrollo de software que se abordarán en esta guía.
- Es importante que el estudiante verifique que todas las herramientas de análisis y pruebas de seguridad (como escáneres de vulnerabilidades, entornos de prueba, etc.) estén configuradas y operativas antes de iniciar las prácticas de la guía.
- Mantener una actitud proactiva y resiliente es crucial, ya que identificar y mitigar vulnerabilidades puede ser desafiante. La perseverancia y una mentalidad orientada a la resolución de problemas son clave en el aprendizaje y aplicación de la seguridad en el desarrollo de software.

NORMAS DE SEGURIDAD

1. Los equipos tecnológicos deben ser utilizados únicamente para fines educativos. No se permite el acceso a páginas web no autorizadas ni la descarga de software no aprobado.
2. Está prohibido fumar, no consumir alimentos y bebidas que pueden dañar los equipos tecnológicos.
3. Procurar usar protectores de pantalla o filtros de luz para evitar la fatiga muscular.
4. Mantener el laboratorio limpio y ordenado, evitar dejar cables sueltos.
5. El acceso al laboratorio debe estar restringido a estudiantes supervisados.
6. En caso de emergencia y activación de la alarma acatar las directrices dadas por profesores y aplicar los protocolos establecidos.
7. Reportar en caso de daños de equipo o mobiliario para garantizar un correcto funcionamiento y seguridad en estos espacios.

Introducción al Problema de la Seguridad del Software

Según lo señalado por Torres (2021), una vulnerabilidad de seguridad se define como una falla o debilidad presente en un sistema de información, lo cual compromete su nivel de seguridad. Es un "agujero" que puede surgir debido a una configuración incorrecta, ausencia de procedimientos, deficiencias en el diseño o por la omisión de medidas de seguridad durante el desarrollo del software.

Los ciberdelincuentes utilizan las debilidades presentes en los sistemas informáticos, como las fallas en los sistemas operativos o las deficiencias en el código, para obtener acceso no autorizado y llevar a cabo acciones ilegales, como el robo de información sensible o la interrupción de las operaciones normales del sistema.

Las principales causas de la aparición de vulnerabilidades son las siguientes:

Figura 27.
Errores en el software

Nota: Principales errores en el software (López, 2020)

Vulnerabilidades y su clasificación

Según Bautista y Chávez (2022) mencionan que “las vulnerabilidades vienen a ser la inconsistencia de los sistemas, donde estas pueden servir para un cibercriminal o atacante con la intención de afectar negativamente los activos de información” (p. 6).

Figura 28.
Tipos de vulnerabilidad del software

Nota. Tipos de vulnerabilidades del software (López, 2020)

Enumeración y Clasificación de Patrones de Ataque Comunes (CAPEC).

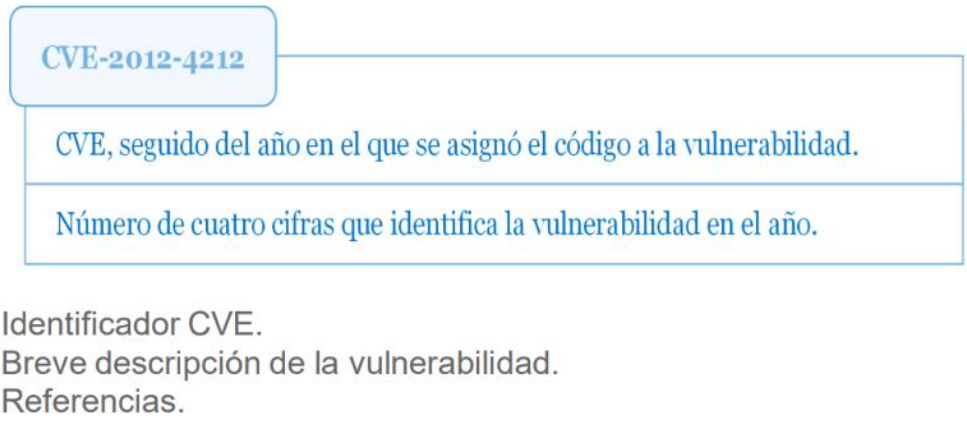
Según MITRE (2023) menciona que (CAPEC) es un recurso estandarizado y exhaustivo que cataloga y describe patrones de ataque comunes en ciberseguridad. CAPEC ofrece un lenguaje común para describir cómo los atacantes explotan las vulnerabilidades en sistemas y software. Cada entrada en CAPEC incluye una descripción detallada del patrón de ataque, métodos, objetivos, técnicas de mitigación y ejemplos. Esta herramienta es fundamental para desarrolladores y profesionales de seguridad, ya que les ayuda a comprender mejor las amenazas y diseñar defensas más efectivas. CAPEC se utiliza en varias fases del ciclo de vida del desarrollo de software, como el modelado de amenazas, diseño de seguridad y pruebas. Además, se relaciona estrechamente con otros estándares de seguridad como CWE y CVE, contribuyendo así a una comprensión más integral de la seguridad cibernética.

Para más información consultar la tabla de vulnerabilidades en: https://www.cvefind.com/en/capec-category.html

Vulnerabilidades y Exposiciones Comunes (CVE).

Según Trujillo y German (2022) dicen que “Actualmente el **CVE** de Mitre es el estándar de facto usado para representar la Información de las vulnerabilidades, y es usada en casi todas las bases de datos como la de National Vulnerability Database (NVD) de NIST. Generalmente, un CVE viene definido por un identificador de la forma CVE–⟨año⟩–⟨identificador⟩”

Figura 29.
Conceptos CVE

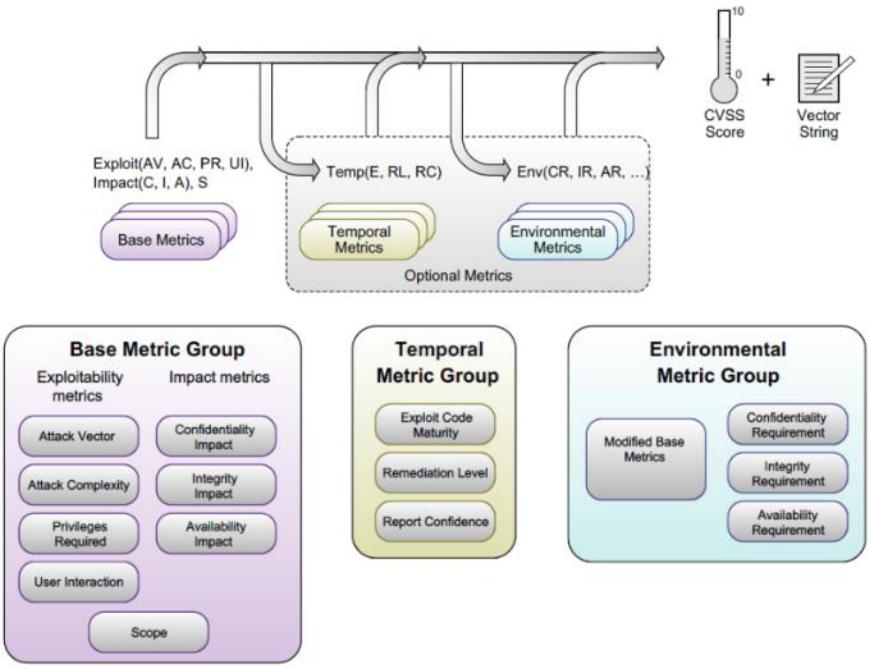


Nota. Conceptos del CVE (López, 2020)

Sistema de Puntuación de Vulnerabilidades Comunes (CVSS).

Es un framework abierto y universalmente utilizado que establece unas métricas para la comunicación de las características, impacto y severidad de vulnerabilidades que afectan a elementos del entorno de seguridad IT. A su vez Incibe (2015) menciona que "CVSS es un sistema de puntuación que proporciona un método estándar y abierto para estimar el impacto de una vulnerabilidad y que se compone tres grupos principales de métricas: Base, Temporal y de Entorno (Environmental)."

Figura 30.
Conceptos de CVSS



Nota. Aspectos del CVSS (Incibe, 2015)

Enumeración de Debilidades Comunes (CWE)

El Common Weakness Enumeration es una lista completa de más de 800 errores de programación, errores de diseño y errores de arquitectura que pueden conducir a vulnerabilidades explotables, más que solo los 25 principales.

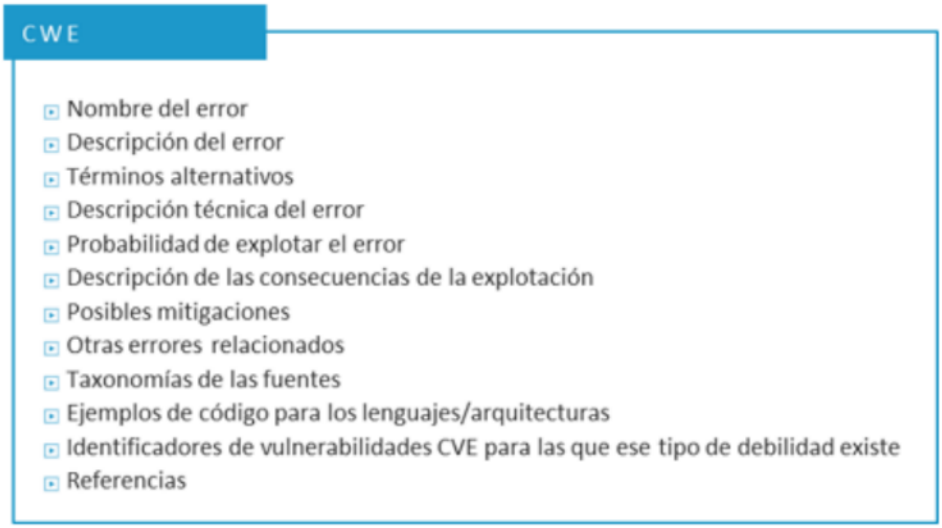
Según Hernández (2020) menciona que el CWE:

Corresponde a una lista o diccionario sobre las deficiencias comunes del software, que pueden generarse en su arquitectura, en el diseño, en el código o en la aplicación.

La CWE fue creado con los siguientes propósitos:

- Servir como un lenguaje común para puntualizar las debilidades de seguridad del software.
- Ser un instrumento de medida estándar para herramientas de software de seguridad orientada a mitigar estas debilidades.
- Brindar un estándar de referencia común para la tipificación de la debilidad, la mitigación y los esfuerzos de prevención. Las debilidades del software son defectos, fallas, vulnerabilidades y otros errores en su aplicación, código, diseño o arquitectura, que si no se resuelven podrían dar lugar a sistemas y redes vulnerables a los ataques (p. 30).

Figura 31.
Conceptos CWE

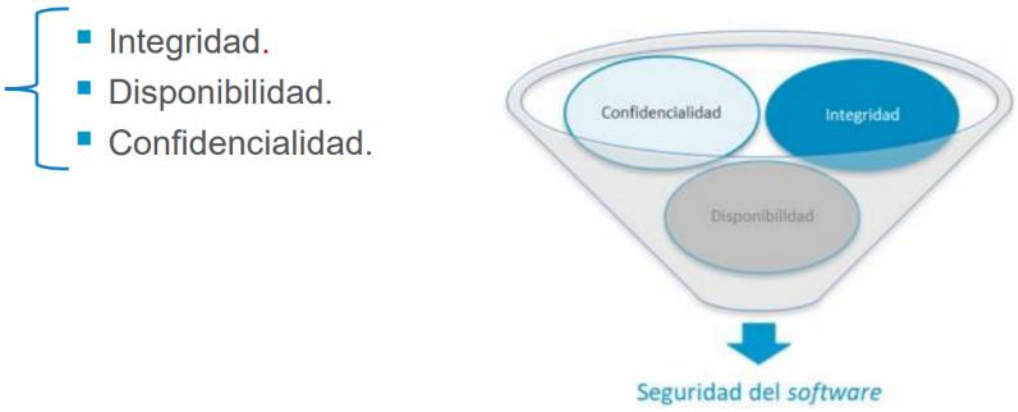


Nota. Conceptos principales de CWE (López, 2020)

Propiedades de un software seguro

Propiedades esenciales

Figura 32.
Propiedades esenciales del Software



Nota. Propiedades esenciales del Software (Bermejo, 2019).

Propiedades complementarias

Figura 33.
Propiedades complementarias del Software

Nota. Propiedades complementarias del Software (Bermejo, 2019).

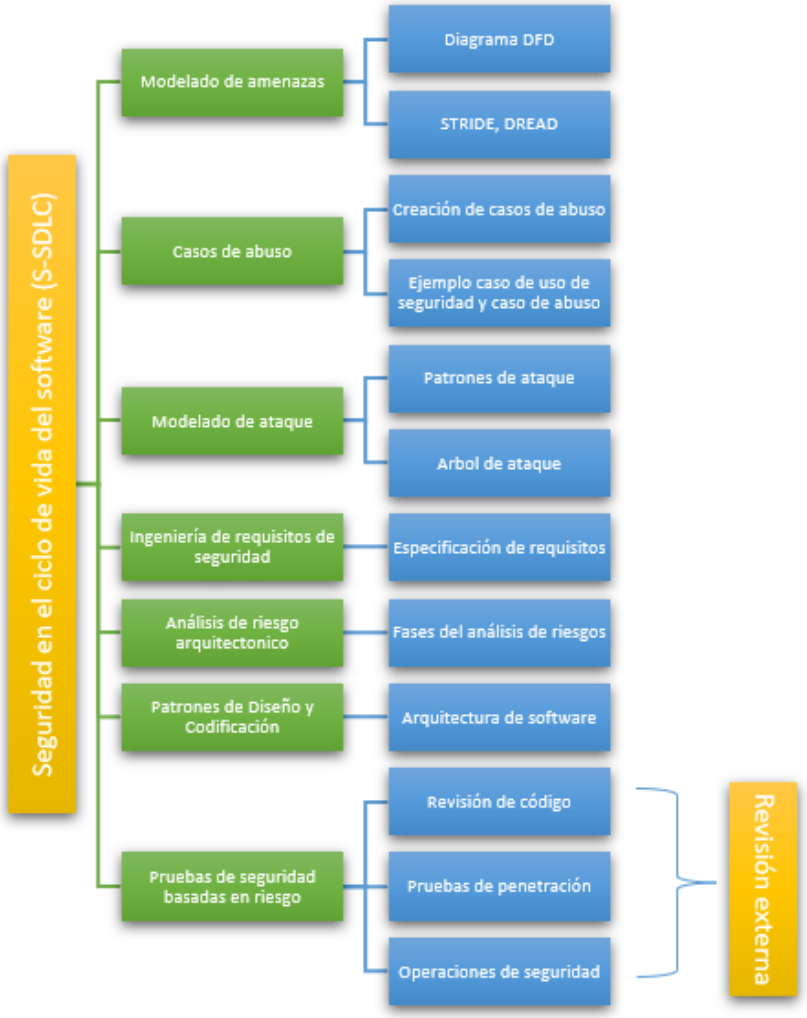
Fundamentos del Ciclo de Vida del Software Seguro (S-SDLC)

El ciclo de vida del desarrollo de software seguro establece una serie de etapas o procesos que un software debe seguir para reforzar su seguridad. Esto permite que el software cumpla con los requisitos del usuario final y asegure la creación de un producto de alta calidad.

Según Trujillo y Chávez (2016) mencionan que:

> Intentar proveer al cliente un sistema seguro en un 100% puede constituir una meta irreal y un tanto utópica, teniendo en cuenta que a medida que estas técnicas para el fortalecimiento han evolucionado, la tecnología así mismo, ha proveído un desarrollo y entendimiento más amplio a personas inescrupulosas para el fortalecimiento de procedimientos que pueden perjudicar el software final (p. 20).

Figura 34.
Fundamentos del S-SDLC



Nota. Fundamentos del S-SDLC Basada en (López, 2020).

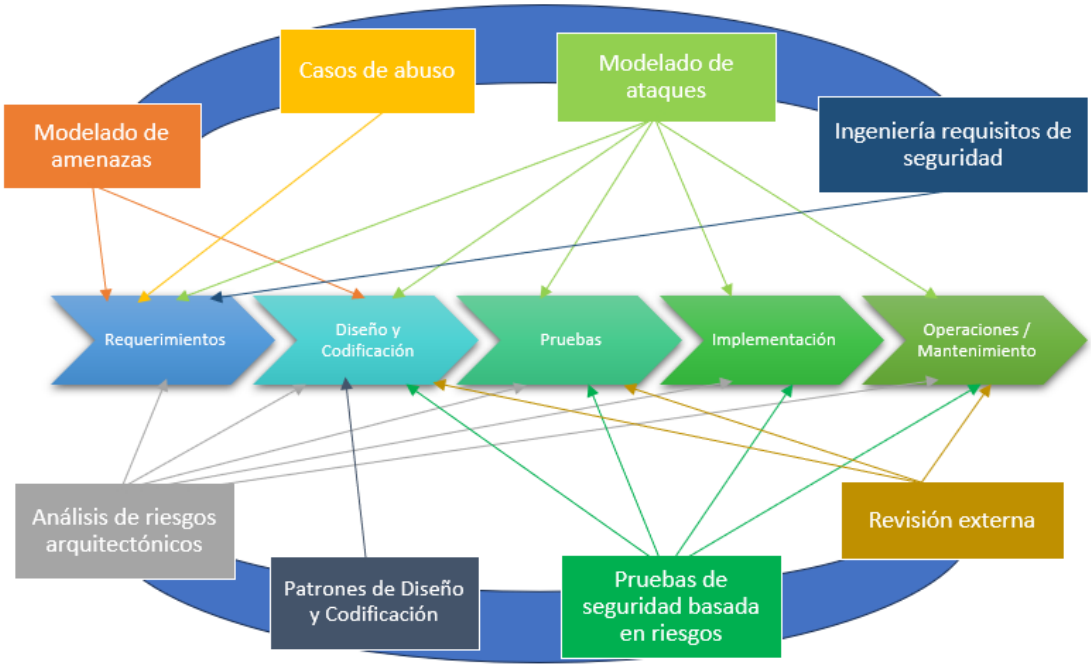
A su vez se puede correlacionar a las prácticas de seguridad con el ciclo de vida del software como se ve en la siguiente tabla:

Tabla 22.
Relación entre las prácticas de seguridad y las fases del ciclo de vida

Fases del SDLC / ***Prácticas de Seguridad***		**Req.**	**Dise.**	**Codif.**	**Prueb.**	**Imple.**	**Ope. / Mant.**
Modelado de amenazas		X	X				
Caso de abuso		X					
Modelado de ataques		X	X	X	X	X	X
Ingeniería requisitos de seguridad		X					
Análisis de riesgos arquitectónicos		X	X	X	X	X	X
Patrones de Diseño y Codificación			X	X			
Pruebas de seguridad basada en riesgos	Revisión de código			X			
	Pruebas de penetración					X	X
	Operaciones de seguridad						X
Revisión externa				X	X		X

Nota. Correlación de las prácticas de seguridad y el ciclo de vida del software basado en (Bermejo, 2019).

Figura 35.
Ciclo de vida del Software Seguro.

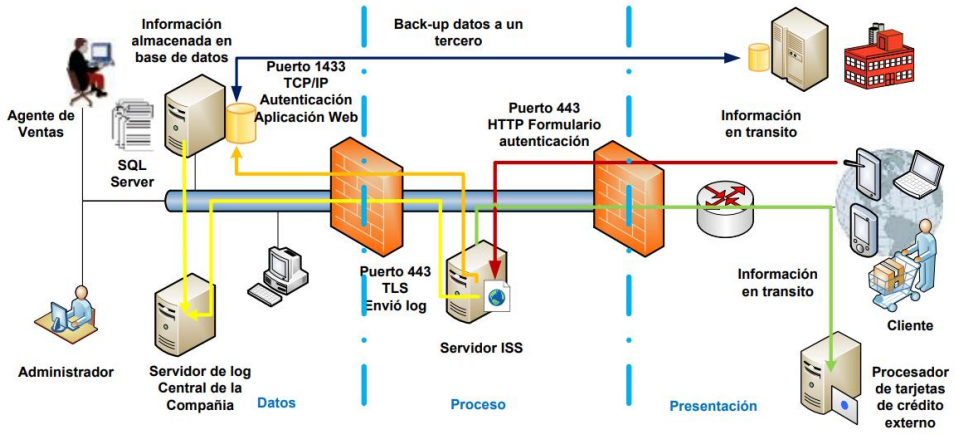


Nota. Ciclo de vida seguro del software (S-SDLC).

Ejercicio práctico

La aplicación permite analizar las amenazas de una aplicación web típica de negocio de pago electrónico de una librería (textos, libros, revistas, etc.), en formato digital con opciones de impresión. El sistema está basado en una típica arquitectura de una aplicación web de tres capas con la siguiente arquitectura lógica:

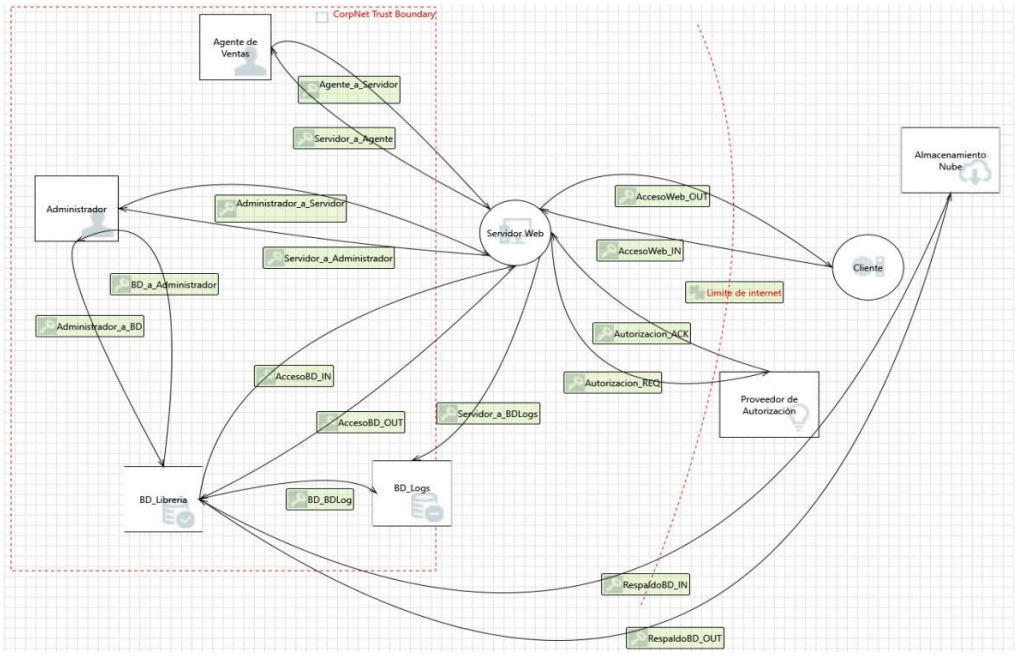
Figura 36.
Ejemplo Arquitectura aplicación web pago electrónico librería.



Nota. Ejemplo de un esquema de una página web de pagos electrónicos de una librería (Ortiz, 2023).

Diagrama DFD del ejemplo:

Figura 37.
Diagrama DFD de la librería.



Nota. Ejemplo del DFD de la librería (Ortiz, 2023)

Fundamento 1

Modelado de amenazas

Una amenaza para una aplicación de software es cualquier actor, entidad, situación o evento que tiene la capacidad de causar daño a los datos o recursos a los que la aplicación tiene acceso o permite acceso. Es importante no confundir este concepto con el de vulnerabilidad, que se refiere a las debilidades que pueden ser explotadas por dichas amenazas.

Las amenazas se pueden clasificar según su intencionalidad:

- Involuntaria.
- Intencional, pero no malicioso.
- Maliciosa.

  Para entender mejor el modelo de amenazas se hace uso de dos videos guías explicados por Bermejo (2019):

- https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=ef5d6c46-b2cf-4f29-aedd-adf700cff411, En este vídeo (Modelado de Amenazas) se presenta y estudia la buena práctica de seguridad Modelado de Amenazas, a desarrollar en las fases de análisis y diseño de un ciclo de vida de desarrollo de un software (SDLC).
- https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=39afa698-2ea6-4dc9-9a5f-adf700cff387, En este vídeo (Microsoft Threat Modelling Tool) se realiza una demostración práctica de la herramienta de modelado de amenazas para aplicaciones Microsoft Threat Modelling Tool.

Diagrama de Flujo De Datos

Lo crucial en esta fase es el Diagrama de Flujo de Datos (DFD). Como se suele decir, "una imagen vale más que mil palabras", y un diagrama puede proporcionar toda la información esencial necesaria. El Diagrama de Flujo de Datos (DFD) es una representación visual del movimiento de datos dentro de un proceso o sistema. Este diagrama es una de las herramientas principales para modelar los procesos de un sistema.

Al crear un diagrama de flujo de datos, es fundamental conocer los elementos que lo conforman y entender su diseño. A mediados de los años 70, Ed Yourdon y Larry Constantine publicaron un libro titulado “Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design”, donde definieron los componentes del DFD y las simbologías correspondientes como:

Figura 38.
Diagrama de flujo

Nota. Simbología del DFD (Ortiz, 2023)

Un diagrama de flujo de datos (DFD) incluye varios componentes clave:

E**ntidades externas**, que son elementos fuera del sistema que envían o reciben información, como empresas o clientes;

Procesos, que transforman los datos de entrada en salidas, por ejemplo, realizar cálculos o dirigir el flujo de información;

Almacenamientos de datos, que retienen la información para su uso futuro, como bases de datos;

Flujos de datos, que representan las rutas que sigue la información entre entidades, procesos y almacenes, cada uno acompañado de una etiqueta descriptiva que clarifica su función dentro del sistema.

Método STRIDE

STRIDE es un acrónimo de seis categorías de amenazas:

Suplantación de identidad (Spoofing), manipulación de datos (Tempering), amenazas de repudio (Repudio), divulgación de información (Information disclosure), denegación de servicio (Denal of service) y elevación de privilegios (Elevation of privileges). Dos ingenieros de Microsoft, Loren Kohnfelder y Praerit Garg, desarrollaron STRIDE.

STRIDE tiene como objetivo asegurar que una aplicación o sistema cumpla con los principios de la tríada de la CIA (confidencialidad, integridad y disponibilidad). Este modelo fue diseñado para que los desarrolladores de software de Windows consideraran posibles amenazas desde la fase de diseño.

Tabla 23.
Tabla de modelo de categorización de amenazas con STRIDE

Categoría	Descripción
Spoofing (Suplantación de identidad)	Suplantación de identidad de usuario. Conjunto de tácticas y técnicas que buscan un compromiso de la gestión de identidad, autenticación y autenticidad del sistema.
Tampering (manipulación de datos)	Manipulación no autorizada. Motivación de afectar a la integridad de los elementos del sistema, modificándolos de manera maliciosa.
Repudiation (amenazas de repudio)	Referente a no repudio. Cualidad de un sistema que permite tener certeza y validez sobre la demostración de autoría en una determinada acción.
Information disclosure (divulgación de información)	Exposición de información. Brecha o fuga de información de clasificación interna o superior de la organización.
Denial of service (denegación de servicio)	Denegación de servicio. Ataques que buscan afectar la capacidad del sistema para ofrecer servicio. Estos ataques pueden afectar al servicio de manera temporal o indefinida.
Elevation of privilege (elevación de privilegios)	Escalada de privilegios. Motivación de realizar acciones para las que un usuario no está autorizado originalmente. Elevando el nivel de permisos y disponibilidad sobre los recursos y funcionalidades del sistema.

Nota. Categorización de amenazas con STRIDE (Ortiz, 2023).

Método de puntuación DREAD

Después de identificar la lista de amenazas, el siguiente paso es evaluarlas según el nivel de riesgo que representan. Esto facilita la priorización de las acciones necesarias para mitigar dicho riesgo. Normalmente, un riesgo se cuantifica multiplicando la probabilidad de que la amenaza ocurra por el impacto que tendría.

Riesgo = Probabilidad * impacto

Se utiliza una escala del 1 al 10 para evaluar la probabilidad de que una amenaza ocurra, donde 1 indica una probabilidad muy baja y 10 una probabilidad muy alta. Del mismo modo, el impacto se mide en la misma escala, con 1 representando un impacto mínimo y 10 un impacto máximo. Este enfoque simplificado permite clasificar las amenazas en una escala de 1 a 10, que se puede dividir en

tres categorías de riesgo: alto, medio y bajo. Por ejemplo, si la probabilidad es 10 y el impacto es 7, el riesgo se calcula como: Riesgo = 10 * 7 = 70.

Una amenaza con un riesgo alto debe ser atendida de inmediato, mientras que una con riesgo medio, aunque importante, es menos urgente. Las amenazas de bajo riesgo podrían incluso ser ignoradas dependiendo del costo y esfuerzo necesarios para mitigarlas. El desafío de este método radica en la dificultad de asignar una valoración uniforme del riesgo cuando diferentes personas realizan la evaluación.

El método DREAD, trata de facilitar el uso de un criterio común respondiendo a las siguientes preguntas:

- Damage potential (Daño potencial): ¿Cuál es el daño que puede originar la vulnerabilidad si llega a ser explotada?
- Reproducibility (Reproducibilidad): ¿Es fácil reproducir las condiciones que propicien el ataque?
- Exploitability (Explotabilidad): ¿Es sencillo llevar a cabo el ataque?
- Affected users (Usuarios afectados): ¿Cuantos usuarios se verían afectados?
- Discoverability (Descubrimiento): ¿Es fácil encontrar la vulnerabilidad

Tabla 24.
Tabla método de puntuación con DREAD

Puntuación	Alto (3)	Medio (2)	Bajo (1)
Damage potential (Daño potencial)	El atacante podría ejecutar aplicaciones con permiso de administrador.	Divulgación de información sensible.	Divulgación de información trivial.
Reproducibility (Reproducibilidad)	El ataque es fácilmente reproducible.	El ataque se podría reproducir, pero únicamente en condiciones muy concretas.	Ataque difícil de reproducir.
Exploitability (Explotabilidad)	Un programador novato podría implementar el ataque en poco tiempo.	Un programador experimentado podría implementar el ataque.	Se requieren cierta habilidad y conocimiento.
Affected users (Usuarios afectados)	Todos los usuarios.	Algunos usuarios.	Pocos usuarios afectados.
Discoverability (Descubrimiento)	Existe información pública que explica el ataque.	La vulnerabilidad afecta a una parte de la aplicación que casi no se utiliza.	El fallo no es trivial, no es muy probable utilizarlo para causar un daño.

Nota. Proceso de puntuación con DREAD (Ortiz, 2023).

Fundamento 2

Casos de abuso

Los diagramas de casos de uso son útiles para identificar los requisitos funcionales de una aplicación; sin embargo, no son tan eficaces para determinar los requisitos de seguridad, especialmente aquellos no funcionales o negativos, que se refieren a acciones que el sistema no debería realizar. Para abordar este problema, se ha desarrollado un tipo especializado de casos de uso que se emplea para analizar y especificar las amenazas de seguridad.

Según Xiaohong et al (2015) lo definen como "Un caso de abuso es la inversa de un caso de uso, es decir, una función que el sistema no debe permitir o una secuencia completa de acciones que resulta en una pérdida para la organización" (p. 68).

Los casos de abuso constituyen un excelente medio de análisis de las amenazas que debe afrontar el software. Son apropiados para el análisis y especificación de restricciones, o requisitos negativos, ya que se basan en el uso indebido del sistema. Establecen la base para otros casos de uso de seguridad que proporcionan los medios para contrarrestar o mitigar las amenazas capturadas en los mismos y una manera altamente reutilizable de organizar, analizar y especificar sus requisitos de seguridad.

En la siguiente figura se muestra la relación entre el caso de uso de seguridad y el de abuso asociado.

Figura 39.
Casos de seguridad y de abuso

Nota. relación entre el caso de uso de seguridad y el de abuso asociado (Bermejo, 2019).

Figura 40.
Diagrama casos de seguridad y de abuso

Nota. Ejemplo de casos de seguridad y de abusos (Donald, 2003).

En la siguiente tabla se resumen las diferencias entre los casos de uso de seguridad y los casos de abuso:

Tabla 25.
Diferencia entre los casos de seguridad y de abuso

Características	Casos de abuso	Casos de seguridad
Uso	Analiza y especifica las amenazas a la seguridad y requisitos de seguridad no funcionales	Analiza y especifica los requisitos de seguridad funcionales
Criterio de éxito	Éxito del atacante	Éxito de la aplicación
Producido	Equipo de seguridad	Equipo de seguridad
Usado	Equipo de seguridad y requisitos	Equipo de seguridad y requisitos
Actor externo	Atacante y usuario	Usuarios
Conducido por	Vulnerabilidades de activos y amenazas	Casos de abuso

Nota. Diferencia entre los casos de seguridad y de abuso (Bermejo, 2019).

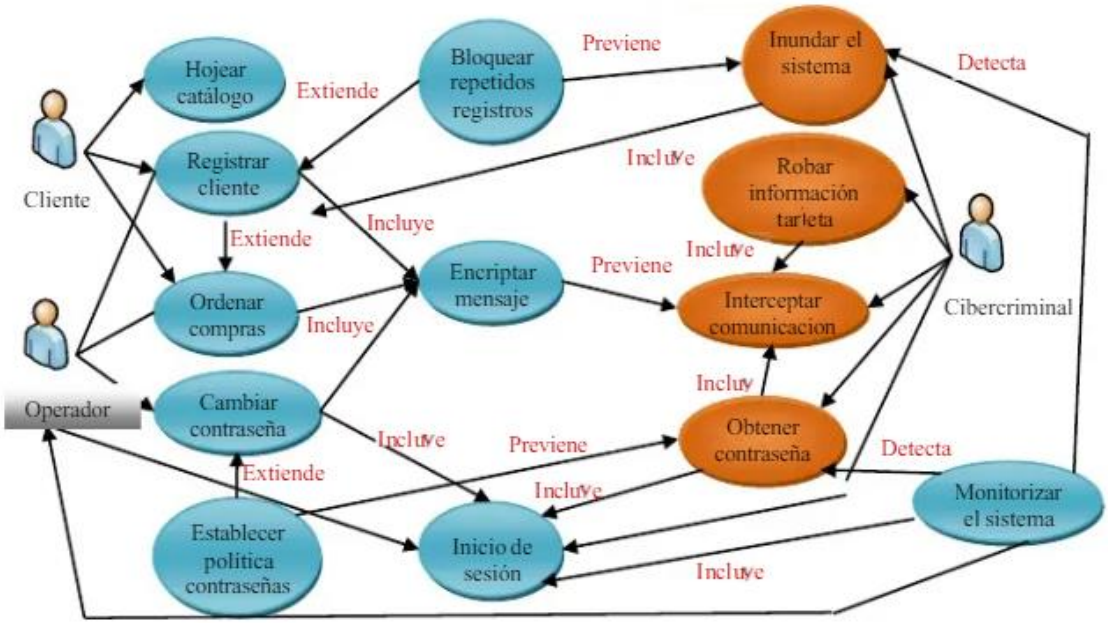
Ejemplo práctico de seguridad y los casos de abuso:

A continuación, presentamos un ejemplo de un diagrama de casos de uso de seguridad y sus casos de abuso asociados en su versión gráfica. La siguiente figura (Sindre y Opdahl, 2001) muestra un caso de abuso para un caso de comercio electrónico que presenta varias relaciones:

- Relación caso abuso: uso seguridad, «incluye» y «extiende».
- Relación caso de uso: abuso, «previene» y «detecta».

Los casos de uso resultantes de especificar los requisitos de seguridad protegen al cajero automático y a sus usuarios de las amenazas potencialmente realizables por un cibercriminal.

Figura 41.
Ejemplo de caso de uso comercio electrónico.



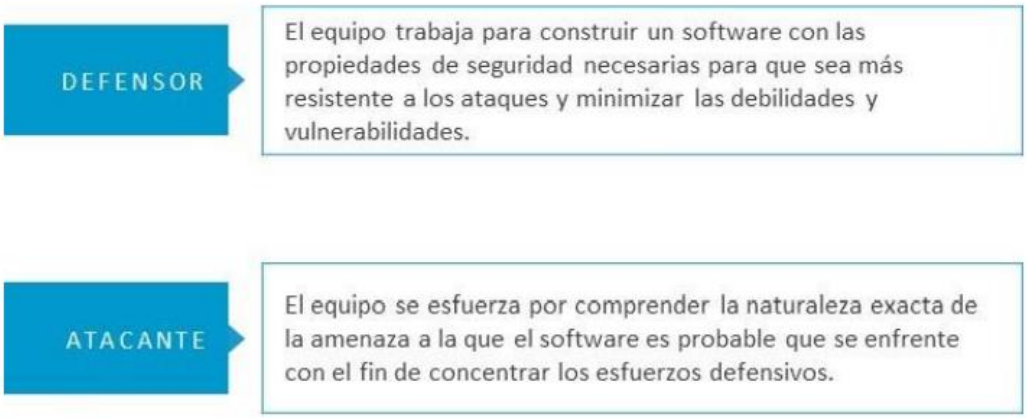
Nota. Ejemplo caso comercio electrónico mediante el caso de seguridad y de abusos (Guttorm y Opdahl, 2001).

Fundamento 3

Modelado de ataques

El objetivo fundamental de la seguridad del software es preservar sus características de seguridad ante ataques perpetrados por individuos malintencionados y minimizar al máximo las vulnerabilidades que puedan ser explotadas. Para asegurar que una aplicación en desarrollo incorpore las propiedades y principios de diseño de software seguro, es necesario que el equipo de diseño y desarrollo adopte dos perspectivas:

Figura 42.
Perspectivas de modelado



Nota. Perspectivas de modelado (Bermejo, 2019).

La perspectiva del atacante se suele modelar de las siguientes dos formas:

- Patrones de ataque.
- Árboles de ataque.

La combinación de patrones de ataque y árboles de ataque permite capturar la probabilidad de cómo los ataques pueden ser combinados y secuenciados, lo que proporciona una serie de datos valiosos para diseñar en el software una serie de respuestas dirigidas a mitigar dichos ataques.

Patrones de ataque

Un ataque aprovecha una vulnerabilidad de una aplicación mediante un exploit para obtener un beneficio del sistema como escalada de privilegios, robo y modificación de datos, modificaciones del funcionamiento, denegación de servicio, etc.

Un patrón de ataque se puede definir como: Un método o herramienta para capturar y representar la perspectiva y el conocimiento del ciberatacante con suficiente detalle sobre cómo se ejecutan los ataques, destacando los métodos de explotación más comunes y las técnicas utilizadas para comprometer el software.

Un catálogo de patrones de ataques, que proporciona un conjunto de definiciones comunes, una taxonomía de clasificación, un esquema de patrones de ataque y un conjunto de ellos, reales, lo constituye la iniciativa del MITRE Common Attack Pattern Enumeration and Classification (CAPEC). Actualmente, incluye 519 patrones reales de ataque. Accede al catálogo a través del aula virtual o desde la siguiente dirección web: http://capec.mitre.org

Tabla 26.
Alcance de un patrón de ataque

Ítem	Descripción
Severidad	En una escala aproximada típica (muy bajo, bajo, medio, alto, muy alto) de la gravedad del ataque
Descripción	Descripción detallada del ataque incluyendo la cadena de acciones tomadas por el atacante. Podría incluir árboles de ataque.
Prerrequisitos del ataque	Describe las condiciones que deben existir, funcionalidades, características del software y comportamiento que debe exhibir para que un ataque de este tipo tenga éxito.
Probabilidad típica del exploit	Es una escala aproximada (Muy Bajo, Medio Bajo, Alto, Muy Alto). Este campo se utiliza para capturar un valor global promedio típico para este topo de ataque.

Método de ataque	Describe el mecanismo de ataque utilizado por este patrón. Este campo puede ayudar a definir la superficie de ataque requerida por este ataque.
Ejemplos	Contiene ejemplos explicativos o casos demostrativos de este ataque.
Conocimiento y habilidades requeridos del ataque	Describe el nivel de habilidad o conocimiento específico requerido por un agente malicioso. Se codifica con una escala aproximada (bajo, medio, alto), así como un detalle de contexto.
Recursos requeridos	Este campo describe los recursos (ciclos de CPU, direcciones IP, herramientas, etc.) requeridos por un atacante para ejecutar con eficiencia este tipo de ataque.
Métodos de prueba	Describe las técnicas normalmente utilizadas para investigar y reconocer un blanco potencial, determina la vulnerabilidad y prepararse para este tipo de ataque.
Indicadores de un ataque	Describe las actividades, eventos, condiciones o comportamientos que podrían servir como indicadores de que un ataque de este tipo es inminente, está en progreso o ha ocurrido.
Soluciones y mitigaciones	Describe acciones o enfoques que pueden potencialmente prevenir o mitigar el riesgo de este tipo de ataque.
Motivación y consecuencias del ataque	Comprender una selección de una lista enumerada de motivaciones definidas o consecuencias. Esta información es útil para la alineación de patrones de ataque a los modelos de amenaza
Descripción del contexto	Describe el contexto en el que este patrón es relevante y aporta más antecedentes para el ataque
Vector de inyección	Describe, lo más exactamente posible, el mecanismo y formato de un ataque.
Payload	Describe los datos de código, configuración u otros a ejecutar o activar, como parte de un ataque del vector de inyección
Zona de activación	Describe el área dentro del software de destino que es capaz de ejecutar o activar de otro modo la carga útil del vector de inyección de un ataque de este tipo. Puede ser un intérprete de comandos, un código de máquina activa en un buffer, un navegador cliente, una llamada API del sistema, etc.
Impacto de activación del payload	Descripción de los efectos causados por la activación del payload en la confidencialidad, integridad o disponibilidad del software solicitado.
Impacto CIA	Describe el impacto de este patrón de ataque en las características de seguridad estándar como la confidencialidad, integridad y disponibilidad.
Vulnerabilidades relacionadas	Que vulnerabilidad o debilidad puede el ataque explotar. Se referencia a la notación estándar de la industria CWE
Requisitos de seguridad relevantes	Identifica los requisitos de seguridad específicos que son relevantes para este tipo de ataques.
Principios de seguridad relacionados	Identifica los principios de diseño de seguridad que son relevantes para identificación o mitigación de este tipo de ataque
Guías relacionadas	Identifica las directrices de seguridad existentes que son relevantes para la identificación o mitigación del tipo de ataque sucedido.

Referencias	Identifica las directrices de seguridad existentes que son relevantes para la identificación o mitigación del tipo de ataque sucedido.

Nota. Describe el proceso de un patrón de ataque (Bermejo, 2019).

Árboles de ataque

Un árbol de ataque se define como un método sistemático para evaluar la seguridad de un sistema, basado en la combinación y las dependencias de sus vulnerabilidades, las cuales un atacante podría explotar para comprometerlo.

La representación de la estructura de un árbol de ataque se realiza conforme a los dos siguientes tipos:

Textual: sigue una estructura de esquema numérico, el nodo raíz, o la meta, representada en el primer nivel sin sangría y cada objetivo de nivel inferior se enumeran con sangría de una unidad por nivel de descomposición:

Figura 43.
Patrones de ataque forma textual

```
Objetivo: Falsificar una Reserva de vuelos
  1. Convencer al empleado de agregar una reserva falsa
        1.1 Chantaje empleado
        1.2 Amenazar empleado
  2. Acceder y modificar la base de datos de vuelos
        2.1 Realizar una inyección SQL en la página web
        2.2 Iniciar una sesión en la base de datos
              2.2.1 Adivinar la contraseña
              2.2.2 Obtener la contraseña rastreando la red (sniff)
              2.2.3 Robar la contraseña del servidor
                    2.2.3.1 Obtener una cuenta del servidor (AND)
                          2.2.3.1.1 Desbordamiento de búfer
                          2.2.3.1.2 Obtener acceso cuenta empleado
                    2.2.3.2 Explotar condición de carrera acceso perfil
                    protegido
```

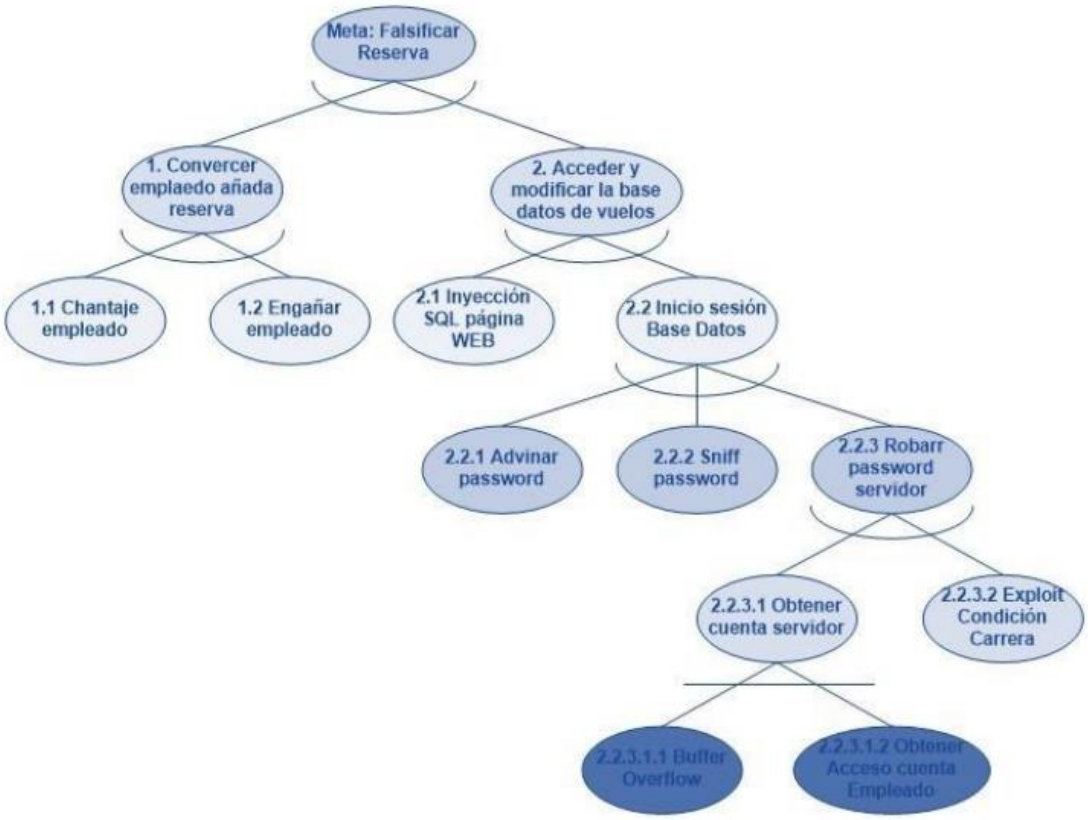
Nota. Ejemplo patrón de ataque forma textual (Bermejo, 2019).

Gráfica o semántica: se construye, generalmente, con el nodo raíz, o la meta, en la parte superior, al descender por las ramas del árbol, se obtienen subobjetivos hasta que se llega al nivel inferior donde están los métodos de ataque. Un nodo se descompone como:

Figura 44.
Patrones de ataque forma gráfica

Nota. Aplicación patrones de ataque a las diferentes fases del SDLC (Bermejo, 2019).

Figura 45.
Árbol de ataque forma gráfica



Nota. Vista conceptual de un árbol de ataque (Bermejo, 2019).

Fundamento 4

Ingeniería de requisitos de seguridad

Muchas de las vulnerabilidades y debilidades del software surgen de requisitos inadecuados, inexactos o incompletos, principalmente debido a una especificación deficiente o insuficiente. Esta falta de precisión no establece adecuadamente las funciones, restricciones y propiedades no funcionales que hacen que el software sea previsible, confiable y resistente.

Según Bermejo (2019) menciona que:

> Los defectos en los requisitos cuestan de 10 a 200 veces más de corregir durante la ejecución que si se detectan durante su especificación. Además, es difícil y costoso mejorar significativamente la seguridad de una aplicación después de que esté en su entorno de producción (p. 26).

La fase de ingeniería de requisitos cubre todas las actividades y tareas que deben realizarse antes de iniciar el diseño, su principal resultado es la especificación de los requisitos que definen los aspectos funcionales y no funcionales del software, en la siguiente figura se muestra un alcance completo de los requisitos de seguridad de una aplicación:

Figura 46.
Requisitos de seguridad

Nota. Especificación de los requisitos de seguridad (Bermejo, 2019).

Requisitos servicios de seguridad (funcionales o positivos). Incluye la especificación de funciones que implementan una política de seguridad, como control de acceso, autenticación, autorización, cifrado y gestión de claves. Deben especificar:

- Propiedades que el software debe exhibir, por ejemplo: el software debe tener un comportamiento correcto y predecible y disponer de capacidades de recuperación frente a ciberataques.
- Nivel requerido de seguridad y salvaguardas de riesgo de las funciones de seguridad.

Requisitos de software seguro (no funcionales o negativos). Requisitos que afectan directamente a la probabilidad de que el software sea seguro. Estos abarcan, principalmente, los no funcionales, los que garanticen que el sistema seguirá siendo confiable, incluso cuando esa confianza se vea amenazada.

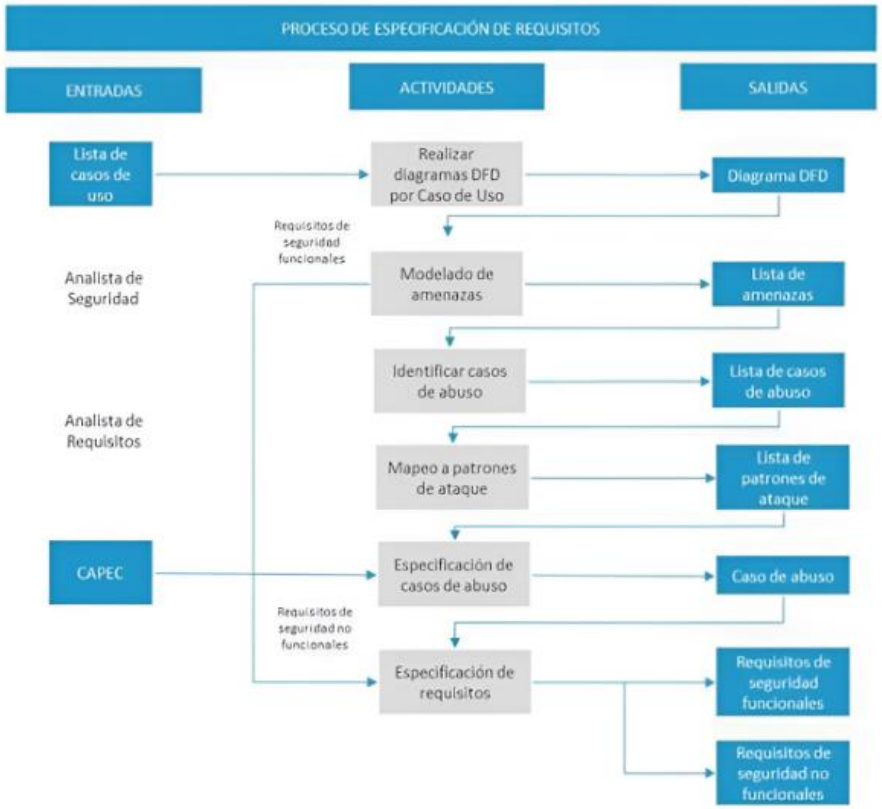
Operacionales. Más enfocados a los controles y normas que rigen los procesos de desarrollo, implementación y operación del software.

Es fundamental que los requisitos de seguridad del software sean:

- Completos.
- Precisos.
- Coherentes.
- Trazables.
- Verificables

Teniendo en cuenta las prácticas de seguridad anteriormente presentadas, los pasos a realizar para la especificación de los requisitos de seguridad se explican a continuación:

Figura 47.
Proceso de especificación de requisitos



Nota. Diagrama del Proceso de especificación de requisitos (Bermejo, 2019).

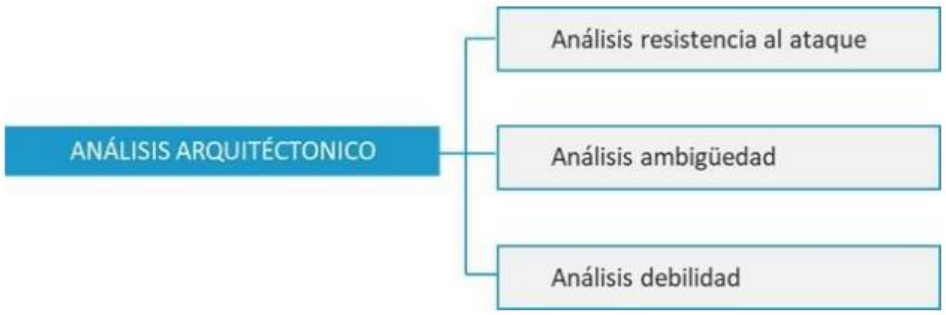
Fundamento 5

Análisis de riesgos arquitectónicos

Según Sanches (2019) menciona que alrededor del 50% de los defectos de seguridad son causados por errores en la fase de diseño, conocidos como debilidades o flaws en inglés. La identificación y gestión de estos riesgos permite mitigar una gran parte de los problemas de seguridad en el software.

El proceso de análisis de riesgo arquitectónico de Cigital usa el proceso de análisis de riesgo arquitectónico que desarrolla un acercamiento a un análisis de riesgo que implica tres pasos básicos.

Figura 48.
Modelo de Cigital, Building Security



Nota. Modelo arquitectónico (Bermejo, 2019).

Análisis de resistencia al ataque. Realiza un modelado de amenazas del diseño completo de la aplicación para analizar la resistencia al ataque. Para ello comprueba una lista de categorías de riesgos según el modelo de análisis de amenazas de Microsoft STRIDE.

Análisis de ambigüedad. Es un subproceso que pretende descubrir nuevos riesgos en base al conocimiento de principios de diseño. Aprovecha múltiples puntos de vista de la arquitectura de software, de varios arquitectos, para crear una técnica de análisis crítica con el fin de encontrar nuevos defectos, puntos de conflicto y de inconsistencia.

Análisis de debilidad. Es un subproceso que ayuda al entendimiento del impacto de dependencias del software externo.

Fundamento 6

Diseño y Codificación

Durante las fases de **Diseño** y **Codificación**, se deben tomar decisiones clave que fortalezcan la seguridad de la arquitectura de software. A continuación, se presentan algunos consejos sobre las mejores prácticas y enfoques centrados en la seguridad para estas fases del SDLC.

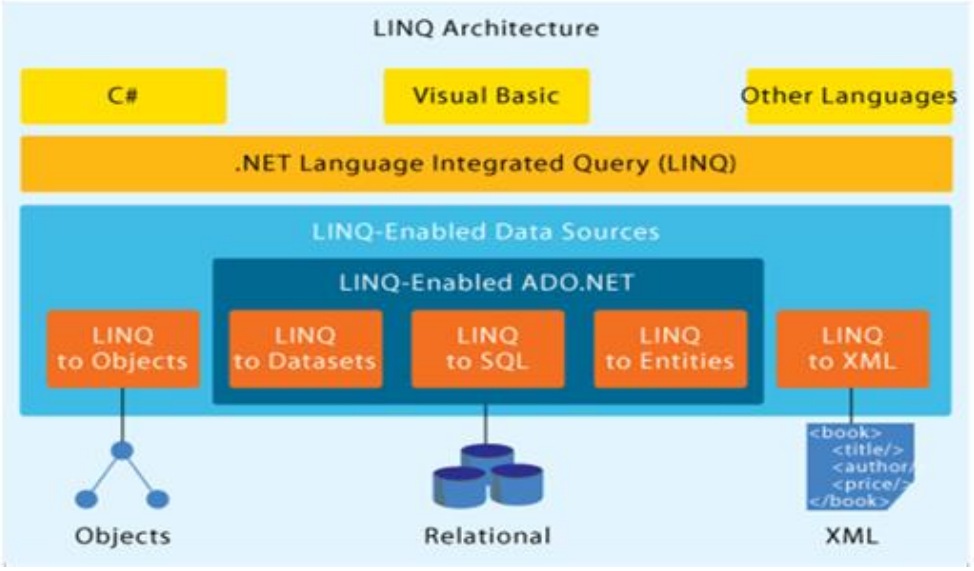
Selección de la Arquitectura de Software Apropiada

- **Arquitectura en Capas con Seguridad Integrada**: Diseñar la aplicación en capas con controles de seguridad en cada nivel. Por ejemplo, en una arquitectura en capas típica, las validaciones de entrada pueden ocurrir en la capa de presentación, la autenticación y autorización en la capa de lógica de negocio, y el cifrado de datos en la capa de acceso a datos.
- **Arquitectura de Microservicios con Seguridad Independiente**: Asegurarse de que cada microservicio tenga sus propios mecanismos de seguridad, como autenticación, autorización y encriptación, para evitar que una vulnerabilidad en un servicio afecte a todo el sistema.
- **Arquitectura de Cliente-Servidor (Client-Server Architecture):** Una estructura donde los clientes hacen solicitudes a un servidor centralizado, que procesa esas solicitudes y devuelve una respuesta. Ejemplo: Un servidor web que sirve páginas HTML en respuesta a las solicitudes HTTP de los navegadores de los usuarios.
- **Arquitectura de Confianza Cero (Zero Trust Architecture):** Asume que las amenazas pueden estar tanto dentro como fuera de la red y que ninguna entidad, ya sea interna o externa, debe ser confiada por defecto. Se aplica la autenticación continua, la segmentación de la red y el monitoreo continuo. **Ejemplo**: Un sistema corporativo donde cada acceso a recursos críticos

requiere autenticación multifactor (MFA) y las conexiones entre microservicios están encriptadas y autenticadas.

- **Arquitectura Orientada a Servicios con Seguridad Integrada (SOA with Integrated Security):** En una arquitectura SOA, cada servicio incluye sus propios mecanismos de seguridad, como autenticación, autorización, y encriptación, asegurando que los servicios sean seguros de manera independiente. **Ejemplo**: Una plataforma de banca en línea donde los servicios de transferencia, consulta de saldo, y gestión de cuentas tienen controles de acceso específicos y utilizan tokens encriptados para la comunicación.
- **Arquitectura LINQ**: facilita la interacción con bases de datos SQL Server. Permite mapear las tablas de la base de datos a clases de objetos en C#, facilitando la manipulación y consulta de datos de una manera más natural y orientada a objetos

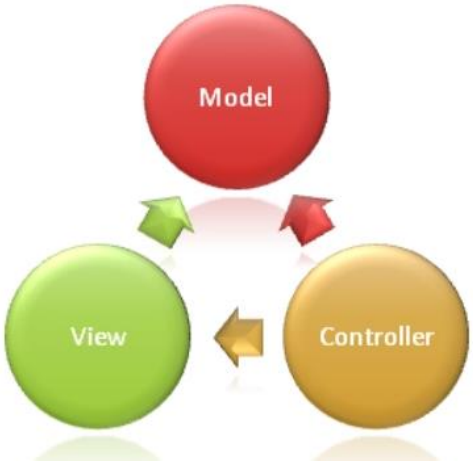
Figura 49.
LINQ arquitectura



Nota. Arquitectura de LINQ (Mayorga & Carrera, 2010).

- **Patrón arquitectónico de Modelo-Vista-Controlador (MVC):** ampliamente utilizado en el desarrollo de aplicaciones, especialmente en aplicaciones web. MVC significa Model-View-Controller (Modelo-Vista-Controlador) y se utiliza para separar las preocupaciones en una aplicación, dividiéndola en tres componentes principales

Figura 50.
Estructura MVC



Nota. marco de MVC (Microsoft, 2024).

Fundamento 7

Pruebas de seguridad basadas en riesgos

Bermejo (2019) menciona que, hasta hace algunos años, las pruebas de software se realizaban con el objetivo de demostrar el cumplimiento de los requisitos, verificando así el correcto funcionamiento de sus funcionalidades y servicios de seguridad. No obstante, estas pruebas no eran suficientes para evaluar cómo se comportaría el software en condiciones anómalas y hostiles, ni para garantizar que estuviera libre de vulnerabilidades (p. 6).

Figura 51.
Tipos de pruebas de seguridad del software

TIPOS DE PRUEBAS SEGURIDAD DEL SOFTWARE

PRUEBAS DE SEGURIDAD FUNCIONALES	¿Cuál es el daño que puede originar la vulnerabilidad si llega a ser explotadas?
PRUEBAS DE SEGURIDAD PERSPECTIVA ATACANTE	¿Es fácil reproducir las condiciones que propicien el ataque?

Nota. Pruebas de seguridad del software (Bermejo, 2019).

Los objetivos de las pruebas de seguridad basadas en el riesgo son los siguientes:

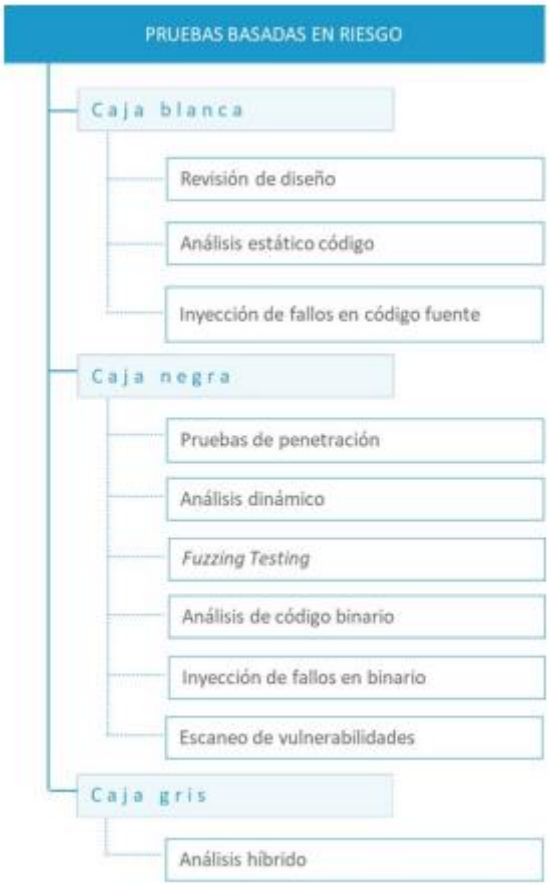
- Verificar la operación confiable del software bajo condiciones hostiles de ataque. Verificar la fiabilidad del software, en términos de comportamiento seguro y cambios de estado confiables.

- Verificar la falta de defectos y debilidades explotables.
- Verificar la capacidad de supervivencia del software ante la aparición de anomalías, errores, y el manejo de estas mediante excepciones, que minimicen el alcance e impacto de los daños que puedan resultar de los ataques.

Las pruebas de seguridad deberían comenzar a nivel de componente antes de la integración del sistema. Se deben de utilizar los modelos de ataque, casos de abuso y análisis de riesgos desarrollados al comienzo del ciclo de vida, para mejorar el plan de pruebas con casos diseñados desde el punto de vista de los atacantes basados en los escenarios de abuso. Esto implica tanto pruebas de caja negra como de caja blanca.

Para tener un concepto más claro las pruebas basadas en riesgos se pueden fundamentar en el siguiente diagrama.

Figura 52.
Pruebas basadas en riesgos.



Nota. Diagrama de pruebas basadas en riesgos (Bermejo, 2019).

Para un mejor entendimiento se puede consultar la fuente de Santos (2020) en su video introductorio de tipos de pruebas de software: https://youtu.be/3hPCueKXlq4

Unas herramientas para la exploración de vulnerabilidades se mencionan a continuación:

Tabla 27.
Herramientas de explotación de vulnerabilidades.

Herramientas de explotación de vulnerabilidades			
Herramienta	**Licencia**	**S.O.**	**Link**
Metasploit Framework	Comercial / Libre	Windows, Linux	https://www.metasploit.com/
Core Impact	Comercial	Windows, Linux	https://www.coresecurity.com/products/core-impact
Canvas	Comercial	Windows, Linux	https://www.siclabs.com/canvas/

Nota. Herramientas para la explotación de vulnerabilidades basada en (Bermejo, 2019).

Revisión de código

El análisis estático del código fuente es considerado, según McGraw (2006), como una de las prácticas de seguridad más importantes que deben llevarse a cabo durante el desarrollo de una aplicación. A continuación, se examinarán los tipos y categorías de herramientas disponibles, tanto comerciales como de código abierto, los lenguajes para los que están disponibles, cómo y cuándo deben usarse, y cómo se integran estas herramientas en el proceso de revisión del código.

Los problemas de seguridad de una aplicación pueden ser resultado de dos tipos de errores principales:

- Errores simples que comete el programador por confusión, un lapsus momentáneo, etc.
- Carencias de conocimientos del programador.

Con esto en mente, el análisis estático de código fuente es adecuado para identificar problemas de seguridad por ciertas razones:

- Las herramientas de análisis estático comprueban el código a fondo y coherentemente, sin ninguna tendencia. Un análisis valioso debe ser lo más imparcial posible.
- Examinando el código en sí mismo, las herramientas de análisis estático a menudo pueden indicar la causa de origen de un problema de seguridad, no solamente uno de sus síntomas.
- El análisis estático puede encontrar errores tempranamente en el desarrollo, aún antes de que el programa sea ejecutado por primera vez.
- Cuando un investigador de seguridad descubre una nueva variedad de ataque, las herramientas de análisis estático ayudan a comprobar de nuevo una gran cantidad de código.

Identificar problemas de seguridad es esencial por varias razones: las herramientas de análisis estático examinan el código de manera exhaustiva y coherente, sin sesgos. Un análisis valioso debe ser lo más objetivo posible. Al centrarse en el código mismo, estas herramientas pueden a menudo señalar la causa raíz de un problema de seguridad, no solo sus síntomas. Además, el análisis estático permite detectar errores en las primeras etapas del desarrollo, incluso antes de que el programa se ejecute por primera vez. Cuando un investigador de seguridad descubre un nuevo tipo de ataque, estas herramientas facilitan la revisión de grandes cantidades de código.

Para más información se puede consultar la fuente de Open Text Corporation (2024): https://www.opentext.com/es-es/productos/fortify-static-code-analyzer

A continuación, se listan algunas herramientas, tanto comerciales como de código abierto, que están disponibles para lenguajes como C/C++ y Java, aunque las herramientas comerciales suelen ser compatibles con una mayor variedad de lenguajes. Las herramientas de código abierto son, en muchos casos, proyectos de investigación desarrollados por universidades. Esta sección se enfoca en enumerar algunas de estas herramientas, especificando su tipo de licencia y los lenguajes que pueden analizar.

Herramientas de análisis estático de código fuente:

Tabla 28.
Herramientas de análisis estático de código.

Nombre de la Herramienta	Licencia	Lenguaje	Página Oficial
SCA de Fortify Software de HP	Comercial	C, C++, Java y otros	https://www.hp.com/es-es/services/managed-print-services/print-solutions.html
AppScam de IBM	Comercial	C, C++, Java y otros	https://www.ibm.com/docs/es/rdfa-and-l/9.1.1?topic=code-running-static-analysis-rules
Prevent de Coverity	Comercial	C, Java y otros	https://www.synopsys.com/software-integrity/static-analysis-tools-sast/coverity.html
K8 Inshight de Klocwork	Comercial	C, Java y otros	https://www.perforce.com/support/consulting/klocwork-consulting
VERACODE	SaaS	C, Java y otros	https://www.veracode.com/products/binary-static-analysis-sast
CXSUITE (CHECKMARX)	Comercial	C, Java y otros	https://checkmarx.com/cxsast-source-code-scanning/
CodeSonar de Grammatech	Comercial	C, C++	https://www.grammatech.com/

SATURN de Stanford University	Libre	C	http://saturn.stanford.edu/index.html
BOOP (C) de Graz University of Technology	Libre	C	https://boop.sourceforge.net/
Magic de Carnegie Mellon University	Libre	C	https://www.cs.cmu.edu/~chaki/magic/
Jtest de Parasoft	Comercial	Java	https://www.parasoft.com/
FindBugs de Maryland k	Libre	Java	https://findbugs.sourceforge.net/
Java PathFinder	Libre	Java	https://sourceforge.net/projects/javapathfinder/
Cppcheck	Libre	C, C++	https://cppcheck.sourceforge.io/
Polyspace de Mathworks	Comercial	C, Ada	https://www.mathworks.com/products/polyspace.html

Nota. Herramientas para el análisis estático de código.

Pruebas de penetración

Una vez completada la fase de desarrollo, se procede al despliegue del sistema. En este punto, es necesario llevar a cabo diversas operaciones y actividades de seguridad relacionadas con la puesta en marcha de la aplicación, para que pueda ser utilizada en producción por los usuarios. Las actividades de seguridad en esta fase incluyen la implementación y verificación de la efectividad de las salvaguardas, tanto a nivel de software (como la autenticación) como de hardware (como los firewalls), que fueron identificadas durante el análisis de riesgos en la fase de diseño.

Según Bermejo J. (2019) menciona que:

> La comprobación de la eficacia de las salvaguardas implementadas se realiza principalmente mediante las pruebas de penetración, que tiene como principal misión verificar cómo el software se comporta y resiste ante diferentes tipos de ataque (p. 35).

El plan de pruebas de penetración debe considerar los escenarios más críticos, en los cuales se puedan recrear vectores de ataques e intrusiones que se consideran extremadamente dañinos, como los escenarios de amenazas internas. Este plan debe incluir:

- ✓ La política de seguridad del sistema que se espera cumplir o hacer cumplir.
- ✓ Las amenazas previstas.
- ✓ Los riesgos de seguridad (derivados de casos de abuso, riesgos arquitectónicos y modelos de ataque).
- ✓ Las posibles secuencias de ataques que podrían ocurrir.

Las pruebas de penetración de una aplicación deben enfocarse en los aspectos relacionados con el comportamiento del software, sus interacciones y las vulnerabilidades que no pueden ser

detectadas por otras pruebas. En este video (Pruebas de Penetración en desarrollo de aplicaciones), se explora la buena práctica de seguridad "Pruebas de Penetración" dentro del contexto del desarrollo y pruebas de seguridad de una aplicación.

A continuación, se hace referencia a dos videos uno es de Bermejo (2019) que explica a más detalle cómo aplicar estas pruebas:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=c42d5ade-161a-4fa2-b7a9-adf700cff2b6, este ejemplo es de Cajas (2023) donde se hace un ataque de inyección SQL por medio de sqlmap (herramienta de Kali Linux) a una dirección URL:

https://youtu.be/OExzpR3hqzk?list=PLVqQOOhLeB2lfnROqmA1eBCovx5D4AD_j

Operaciones de seguridad

El proceso final para llevar a cabo, previo al paso a producción de la aplicación segura, se compone de las actividades centrales de:

- ✓ Distribución.
- ✓ Despliegue.
- ✓ Operaciones.

Distribución

El propósito de una distribución segura es minimizar el riesgo de que el software sea accesado o manipulado por agentes maliciosos durante su transmisión del proveedor al consumidor, ya sea mediante el envío físico o la descarga en red. Para asegurar una distribución y despliegue seguros del software desarrollado, se recomienda seguir estas buenas prácticas:

- ❖ Cambio de los valores de configuración predeterminados durante el desarrollo.
- ❖ Utilizar mecanismos de distribución estándar, como los utilizados en los derechos de propiedad intelectual.
- ❖ Distribuir el software con una configuración por defecto segura y lo más restrictiva posible.
- ❖ Realizar una guía de configuración de seguridad.
- ❖ Proporcionar una herramienta de instalación automática.
- ❖ Establecer un medio de autenticación para la persona que va a ejecutar la instalación y configuración.
- ❖ Los interfaces de configuración proporcionados por la herramienta o el script de instalación deben ser seguros.

- ❖ Revisión y limpieza de todo el código fuente por el usuario visible (por ejemplo, código del cliente de aplicaciones web).

Despliegue

Una configuración cuidadosa y la personalización del entorno de despliegue de cualquier aplicación de software pueden mejorar enormemente su seguridad. El diseño de un entorno de despliegue adaptado para una aplicación requiere de un proceso:
Comienza en el nivel de componente de red.

- Continúa por el sistema operativo y otro software de base como puede ser un gesto de base de datos, etc.
- Termina con la propia configuración de seguridad de la aplicación y el sistema.

Figura 53.
Capas del sistema a proteger

Nota. Capas del sistema que debe ser protegido (Bermejo, 2019).

El endurecimiento del software implica el uso de procesos y herramientas para gestionar y corregir defectos y debilidades en las configuraciones del software, basándose en un proceso de control de configuración. En España, el Centro Criptológico Nacional (CCN), y en Estados Unidos, el Departamento de Defensa y otras agencias, desarrollan directrices de configuración segura y listas de verificación para productos de software comercial. Algunas de estas se pueden encontrar en:

- Centro Criptológico Nacional (CCN)
- NIST Listas de comprobación de configuración de productos TIC

Operaciones

Muchos desarrolladores de software sostienen que la distribución, el despliegue y las operaciones no forman parte del proceso de desarrollo. Sin embargo, es crucial ajustar los controles de acceso a la red y a los niveles del sistema operativo, así como diseñar sistemas para el registro de eventos, la monitorización, y el respaldo y recuperación de archivos. Estos elementos serán esenciales para una respuesta eficaz ante incidentes. Los ataques son inevitables, por lo que es necesario estar preparado tanto para defenderse de ellos como para remediar los daños después de que se hayan producido.

A continuación, se da una serie de hosting gratuitos para subir el código fuente de las aplicaciones web con el fin de realizar las pruebas anteriormente mencionadas:

Tabla 29.
Lista de hosting gratuitos.

Proveedor	Descripción	URL	Lenguaje
Azure Free Tier	Hosting gratuito limitado con SQL de 250 MB y 1 GB de almacenamiento.	Azure Free Tier	**ASP.NET**
SmarterASP.NET	60 días de prueba gratuita con soporte para ASP.NET y SQL Server.	SmarterASP.NET	**ASP.NET**
GearHost	Proporciona un plan gratuito con soporte para ASP.NET, 1 base de datos SQL.	GearHost	**ASP.NET**
MyWindowsHosting	Ofrece un plan gratuito para aplicaciones ASP.NET con 1 GB de espacio y 1 base de datos SQL.	MyWindowsHosting	**ASP.NET**
Somee.com	Hosting gratuito con soporte para ASP.NET y SQL Server, con ancho de banda limitado.	Somee.com	**ASP.NET**
Azure Free Tier	Hosting gratuito limitado con SQL de 250 MB y 1 GB de almacenamiento.	Azure Free Tier	**ASP.NET**
Heroku	Soporta aplicaciones Java con 550 horas de uso gratuito al mes y base de datos.	Heroku	**Java**
Red Hat OpenShift	Plan gratuito para despliegue de aplicaciones Java con soporte para contenedores.	Red Hat OpenShift	**Java**
Amazon AWS Free Tier	Ofrece EC2 gratis por 12 meses, ideal para desplegar aplicaciones Java.	Amazon AWS Free Tier	**Java**

Jelastic Cloud	Plataforma en la nube con plan gratuito para aplicaciones Java.	Jelastic	**Java**
Cloud Foundry	Soporte para aplicaciones Java con opciones de hosting gratuito.	Cloud Foundry	**Java**
000webhost	Hosting gratuito para PHP con MySQL, 1 GB de espacio y 10 GB de ancho de banda.	000webhost	**PHP**
InfinityFree	Hosting PHP gratuito con almacenamiento y ancho de banda ilimitado, MySQL.	InfinityFree	**PHP**
FreeHosting.com	Ofrece 10 GB de espacio, 250 GB de ancho de banda y soporte para PHP y MySQL.	FreeHosting.com	**PHP**
AwardSpace	Plan gratuito con 1 GB de almacenamiento, 5 GB de ancho de banda y soporte para PHP.	AwardSpace	**PHP**
Byet.host	Hosting PHP con MySQL gratuito y almacenamiento ilimitado.	Byet.host	**PHP**
000webhost	Hosting gratuito para PHP con MySQL, 1 GB de espacio y 10 GB de ancho de banda.	000webhost	**PHP**

Nota. Lista de hosting para publicación de páginas web.

Fundamento 8

Revisión externa

Un análisis realizado por personal externo al equipo de diseño resulta ser muy eficaz y crucial, ya que proporciona una perspectiva diferente sobre la seguridad del sistema y los riesgos asociados, contribuyendo así a una mejora en la seguridad general. Este análisis externo es probable que identifique amenazas y riesgos residuales que no se habían detectado previamente.

Una vez completado el análisis externo, se debe revisar el análisis de riesgos y, si se identifican nuevas amenazas y riesgos, deben ser gestionados y mitigados. Esto puede requerir modificaciones en la arquitectura de hardware y software, ajustes en el código, y la repetición de las revisiones y pruebas de seguridad según los nuevos riesgos identificados. Finalmente, después del despliegue de la aplicación, se deben realizar nuevamente pruebas de penetración.

Este esquema de seguridad en el ciclo de vida es cíclico, lo que significa que algunas actividades pueden necesitar repetirse varias veces debido a la naturaleza en constante cambio y evolución de las aplicaciones. Además, el ciclo de vida del desarrollo de aplicaciones a menudo implica la refinación de prototipos hasta llegar al sistema final. Una práctica habitual en la adquisición de software es aceptar productos que cumplen con las funcionalidades requeridas sin prestar suficiente

atención a la especificación y aseguramiento de las propiedades de seguridad. Esto incrementa la exposición y los riesgos de seguridad para la organización.

Pana analizar mejor este fundamento nos basaremos en la información que Bermejo (2019) nos proporciona en el siguiente video: https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=7e666bb0-4150-4be9-8932-adf700cff21e

Materiales

- Computador.
- Pizarra.
- Microsoft Office 365 (Teams, Stream, etc).
- Plataforma Virtual.
- IDE's de desarrollo
- Gestores de Base de Datos
- Kali Linux
- Entornos de trabajo controlados
- Laboratorios especializados en desarrollo de software

Bibliografía

Álvarez, G. (24 de septiembre de 2018). *Kyocode*. Ciclo de vida de un software: https://www.kyocode.com/2018/09/ciclo-de-vida-de-un-software/

Bermejo, J. (2019). Desarrollo Seguro de Software y Auditoría de la Ciberseguridad. *UNIR - Electronics, 8*(11), 1218. https://doi.org/https://doi.org/10.3390/electronics8111218

Díaz, G. (2015). *Estudio de Técnicas Automáticas de Análisis de Vulnerabilidades de Seguridad en Aplicaciones Web.* UNED.

Garcia, V. (2020). Metodologías de desarrollo de software seguro con propiedades agiles. *Universidad Laica Eloy Alfaro de Manabí (ULEAM), 5*(10), 1027-1046. https://doi.org/10.23857/pc.v5i10.2658

Hazlegreaves, S. (15 de octubre de 2021). *Education sector suffers series of cyber attacks in 2021. Open Access Government.* https://www.openaccessgovernment.org/education-sector-suffers-series-of-cyber-attacks-in-2021/122465/

Hernández, E. O. (2023). *Modelado de Amenazas de una Aplicación Seguridad en el software.* Universidad UNIR.

Howell, J. (25 de Agosto de 2022). *Threat Modeling Tool feature overview*. https://learn.microsoft.com/es-es/azure/security/develop/threat-modeling-tool-feature-overview

IBM. (10 de mayo de 2024). *IBM*. ¿Qué es el desarrollo de software?: https://www.ibm.com/es-es/topics/software-development

López, Á. D. (2020). Método para el desarrollo de software seguro basado en la ingeniería de software y ciberseguridad. *Universidad ECOTEC, Ecuador, 5*(3.1), 263-280. https://doi.org/10.33890/innova.v5.n3.1.2020.1440

McGraw, G. (2006). ResearchGate. *Berryville Institute of Machine Learning, 17*(1), 7-10. https://doi.org/10.1109/ISSRE.2006.43

Microsoft. (3 de febrero de 2010). *Microsoft*. Simplified Implementation of the Microsoft SDL: https://www.microsoft.com/en-us/download/details.aspx?id=12379

Ortega, J. (2020). Desarrollo seguro en ingeniería del software: Aplicaciones seguras con Android, NodeJS, Python y C++. Alpha Editorial. https://books.google.es/books?id=x3J6EAAAQBAJ&lpg=PA7&hl=es&pg=PA4#v=onepage&q&f=false

OWASP. (19 de enero de 2020). *OWASP Top Ten*. OWASP Foundation: https://owasp.org/www-project-top-ten/#

Pachacuti, M. (2020). *MODELO DE SEGURIDAD PARA EL DESARROLLO DE SOFTWARE [Tesis de maestria, UNIVERSIDAD MAYOR DE SAN ANDRÉS].* Repositorio institucional. https://repositorio.umsa.bo/xmlui/bitstream/handle/123456789/27697/TM-3721.pdf?sequence=1&isAllowed=y

Parraga, J. (marzo de 2024). *Propuesta de plan de seguridad para mitigar vulnerabilidades en el ciclo de integración continua y despliegue de aplicaciones web en DevOps.* http://repositorio.uisrael.edu.ec/handle/47000/4135

Pérez, O., Gainza, D., y Raúl, H. (29 de enero de 2023). *Estudio del desarrollo seguro del software*. https://publicaciones.uci.cu/index.php/serie/article/view/1291

Retena, P. (Junio de 2023). *Análisis y Modelado de Amenazas* . https://prezi.com/uky4fywbiexu/analisis-y-modelado-de-amenazas/

Sampieri, R. H. (2014). Metodología de la investigación. méxico: mcGraw-Hill. https://apiperiodico.jalisco.gob.mx/api/sites/periodicooficial.jalisco.gob.mx/files/metodologia_de_la_investigacion_-_roberto_hernandez_sampieri.pdf

Sierra, T. (2022). *LA SEGURIDAD INFORMÁTICA EN EL DESARROLLO DE APLICACIONES WEB [UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD].* Respositorio institucional. https://repository.unad.edu.co/bitstream/handle/10596/54049/ta52sie605.pdf?sequence=1&isAllowed=y

Trujillo, D., y Chávez, A. (2016). *Sistema de Control de Versiones para el Desarrollo de Software Seguro [Trabajo investigativo, Fundación Universitaria los Libertadores].* Repositorio institucional. https://repository.libertadores.edu.co/server/api/core/bitstreams/416eb844-5e4b-4a06-a44d-c8d65dc654bb/content

Viteri, C. (marzo de 2023). *Propuesta de Políticas de seguridad informática en el desarrollo de sistemas web, aplicando la norma ISO 27002, para la Universidad Central del Ecuador.* http://repositorio.uisrael.edu.ec/handle/47000/4141

Xiaohong, Y., Emmanuel, N., Imano, W., y Huiming, Y. (2015). Developing Abuse Cases Based on Threat Modeling and Attack Patterns. Journal of Software. https://doi.org/10.17706/jsw.10.4.491-498.

ANEXO 5

REPORTE TURNITIN

Trabajo de titulación_LinoCajas_Tesina

INFORME DE ORIGINALIDAD

8%	9%	5%	2%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	www.escolapios-soria.com Fuente de Internet	2%
2	repositorio.uisrael.edu.ec Fuente de Internet	2%
3	hdl.handle.net Fuente de Internet	1%
4	biblio.unvm.edu.ar Fuente de Internet	1%
5	vsip.info Fuente de Internet	1%
6	qdoc.tips Fuente de Internet	1%
7	c3.usac.edu.gt Fuente de Internet	1%

Excluir citas	Activo	Excluir coincidencias	< 1%
Excluir bibliografía	Activo		